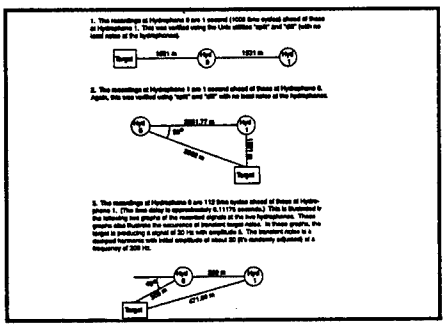
Simulating a Multi-target Acoustic Array on the Intel Paragon

Charles A. Jones

Technical Report No. 108
October, 1994

Center for Computational Statistics



George Mason University
Fairfax, VA 22030

CENTER FOR COMPUTATIONAL STATISTICS
TECHNICAL REPORT SERIES (RECENT REPORTS)

TR 93. Winston C. Chow, Modeling and Estimation with Fractional Brownian Motion and Fractional Gaussian Noise (Ph.D. Dissertation), February, 1994.

TR 94. Mark C. Sullivan and Edward J. Wegman, Correlation Estimators Based on Simple Nonlinear Transformations, February, 1994, To appear **IEEE Transactions on Signal Processing.**

TR 95. Mark C. Sullivan and Edward J. Wegman, Normalized Correlation Estimators Based on Simple Nonlinear Transformations, March, 1994.

TR 96. Kathleen Perez-Lopez and Arun Sood, Comparison of Subband Features for Automatic Indexing of Scientific Image Databases, March, 1994.

TR 97. Wendy L. Poston and Jeffrey L. Solka, A Parallel Method to Maximize the Fisher Information Matrix, June, 1994.

TR 98. Edward J. Wegman and Charles A. Jones, Simulating a Multi-target Acoustic Array on the Intel Paragon, June, 1994.

TR 99. Barnabas Takacs, Edward J. Wegman and Harry Wechsler, Parallel Simulation of an Active Vision Model, June, 1994.

TR 100. Edward J. Wegman and Qiang Luo, Visualizing Densities, October, 1994.

TR 101. Daniel B. Carr, Converting Tables to Plots, October, 1994.

TR 102. Julia Corbin Fauntleroy and Edward J. Wegman, Parallelizing Locally-Weighted Regression, October, 1994.

TR 103. Daniel B. Carr, Color Perception, the Importance of Gray and Residuals on a Choropleth Map, October, 1994.

TR 104. David J. Marchette, Carey E. Priebe, George W. Rogers and Jeffrey L. Solka, Filtered Kernel Density Estimation, October, 1994.

TR 105. Jeffrey L. Solka, Edward J. Wegman, Carey E. Priebe, Wendy L. Poston and George W. Rogers, A Method to Determine the Structure of an Unknown Mixture Using the Akaike Information Criterion and the Bootstrap, October, 1994.

TR 106. Wendy L. Poston, Edward J. Wegman, Carey E. Priebe and Jeffrey L. Solka, A Contribution to the Theory of Robust Estimation of Multivariate Location and Shape: EID, October, 1994.

TR 107. Clifton D. Sutton, Tree Structured Density Estimation, October, 1994.

TR 108. Charles A. Jones, Simulating a Multi-target Acoustic Array on the Intel Paragon (M.S. Thesis), October, 1994.

Simulating a Multi-target Acoustic Array on the Intel Paragon

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Statistical Science at George Mason University.

By

Charles A. Jones
Doctor of Philosophy, Dartmouth College, 1978

Director: Dr. Edward J. Wegman
Dunn Professor of Statistics
Center for Computational Statistics

Summer 1994
George Mason University
Fairfax, Virginia

Acknowledgements

I dedicate this thesis to my parents, Chester and Kathleen Jones. They are wonderful people and they have always encouraged and supported me.

There are many people who made the completion of this thesis and my year of learning statistics rewarding and enjoyable. Foremost, I give my sincerest thanks to Dr. Edward J. Wegman for his guidance throughout my work on this project. I would also like to thank him for awarding me a graduate research assistantship, which allowed me to fully concentrate my efforts on learning statistics and on this thesis. In addition, Drs. John J. Miller, Clifton D. Sutton, and Daniel B. Carr taught excellent courses, which provided me with an outstanding foundation in statistics.

I also thank Duane King, Jim Turtora, and Qiang Luo for their outstanding administration of the computer systems and their prompt attention to my individual questions and requests.

Finally, I would like to thank Anita, Ben, and Daniel for being a marvelous family. Their love and support mean the world to me.

The computational effort for this research was performed on an Intel Paragon, which was purchased under contract N00014-93-I-0527 with the Office of Naval Research. My research assistantship was funded under an AASERT grant attached to the Army Research Office Contract DAAL03-91-G-0039.

Table of Contents

List of Figures

Abstract

SIMULATING A MULTI-TARGET ACOUSTIC ARRAY ON THE INTEL PARAGON

Charles A. Jones, Ph.D.

George Mason University, 1994

Thesis Director : Dr. Edward J. Wegman

This thesis is built around a parallel programming project which simulates the recordings of a linear array of hydrophones in the presence of several sources of acoustic signals (targets). Simulation of multi-target data with appropriate modeling of multipath, path refraction, and local and distant noise sources is a useful and relatively sophisticated modeling chore. The signal suites include multiple frequency narrow band signals, broad-band flow noise, and randomly generated transient signals. The noise suites include coherent and incoherent Gaussian noise, impulsive noise, and continental shelf reverberation.

This project is of interest from a parallel programming viewpoint because it uses the Paragon as a true Multiple Instruction Multiple Data (MIMD) machine with three types of nodes: 1) a Manager node, 2) target nodes, and 3) hydrophone nodes. The target nodes are responsible for generating the signal suites, transient noise, and the coherent noise. The hydrophone nodes are responsible for calculation of multipath, refraction and time delay as well as adding the local incoherent noise suite. While this simulation can be used as a stand-alone application, it also will form the foundation for a much larger, more sophisticated simulation, namely producing a virtual Command Information Center (CIC). The larger simulation

will involve the hydrophone nodes directly passing their simulated recordings to array processing nodes running beam forming and other signal processing algorithms in an attempt to locate and identify the simulated targets. The output from these nodes will then be sent via a HIPPI Channel to our Virtual Reality Lab, where the information will be graphically displayed, thus producing a virtual CIC.

Chapter 1

Introduction

1.1 The Physical Situation being Simulated

This project simulates a linear array comprised of up to 30 acoustic hydrophones and up to 15 targets. Each target produces a suite of narrow-band harmonic signals. These signals are produced by rotating equipment on board the target. In addition, at each target the project simulates broad-band flow noise and transient noise. The flow noise is due to the movement of the target through the water and propulsion noise. The transient noise simulates infrequent events such as ballast operations or hatches being opened.

Each of the hydrophones records the signals it receives from all targets and, more predominantly, ambient noise. This noise is modeled with two components: incoherent white noise (Gaussian) and occasional, loud, local, short duration, impulsive noise. The white noise component reflects continual ambient sounds and any additions to the signals due to the hydrophone itself. The impulsive noise models local noise such as animal sounds near the hydrophone. This project simulates each hydrophone recording 1000 observations per second.

While the hydrophones are arranged in a straight line, the targets are arranged arbitrarily. (See figure 1.1.)

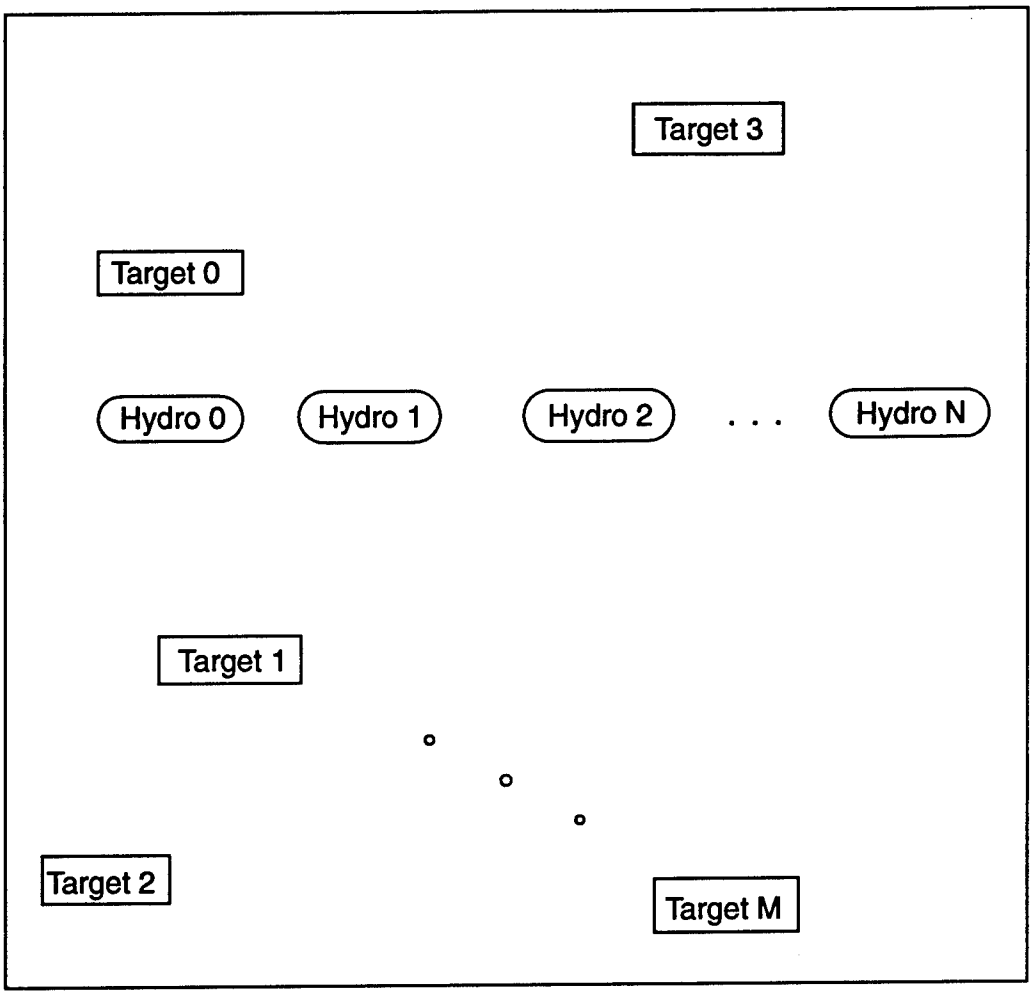


Figure 1.1: Sample Layout of Positions of Targets and Hydrophones

1.2 A Brief Overview of Signal Processing

This simulation produces a discrete sequence of real numbers at each of the hydrophone nodes. These sequences are meant to mimic the discrete-time sampling by actual hydrophones of the continuous-time signals being received from the targets. There is a vast literature on the general problem of discrete-time, digital, signal processing, see, for example, Oppenheim and Schafer [2] and Stearns and David [4]. The purpose of this section is to outline some types of signal processing procedures that can be undertaken to recover the location and classification of the targets based on the hydrophones' sampling. See Pillai [3], Hall [1], Wegman and Smith [5], and Wegman, Schwartz, and Thomas [6] for detailed discussions of multi-sensor signal processing.

Univariate Procedures

Univariate procedures fall into three categories: detection, estimation, and classification.

Detection procedures, as the name implies, attempt to detect the presence of a target signal in the midst of the ambient noise. Typically, they consist of a statistical hypothesis test with null hypothesis: there *is* a target signal embedded in noise. The alternative hypothesis is: there is just noise. Thus, the status-quo is the presence of a target. The burden of proof is on the statistical procedure to give strong evidence of no target. This choice of null and alternative hypotheses allows the type I error rate (which can be set at any desired level) to correspond to the probability of missing an existing target. Nondetection of an existing target can obviously be quite serious from a tactical standpoint; this hypothesis scheme gives control of the probability of making such an error. On the other hand, in this scheme, type II errors correspond to false alarms. While not as serious individually, large numbers of these errors could adversely effect the quality of the response to all detections (false or not). As is usually the case, statistics

illuminates the consequences of decisions that must be made.

Estimation procedures traditionally attempt to determine the frequencies and amplitudes of the narrow-band portions of a signal, given that a target exists. A standard assumption about a target signal is that it contains a narrow-band frequency suite consisting of sinusoids. These sinusoids are produced by rotating equipment; examples of such equipment are electrical generators, winches, air conditioning blowers, engines, transmissions, and propeller shafts. Good estimates of these frequency suites are vital to identifying the target. In addition, the frequencies of the drive-train equipment can give an estimate of the speed at which the target is moving. The obstacle to determining these frequencies is the presence of noise. Traditionally, the noise is assumed to be Gaussian distributed with mean $= 0$ and constant variance. The basic estimation technique is to perform regression using trigonometric polynomials.

Classification procedures attempt to identify the target based on the signal analysis. Examples of types of identification would be to determine whether the target is a friend or foe, and more specifically whether that the target belongs to a specific class of submarine. The classic example of a friend or foe identification technique is to use the frequency of the electrical generator; United States ships operate with 60 Hertz generators, while almost all other countries use 50 Hertz. In general, these procedures involve data base and pattern matching techniques.

Multiple Sensor Array Processing

The standard technique in the multiple target situation when there is an array of hydrophones is called *beam forming.* The basic concept of this technique is to average the sampled signal value from each hydrophone using time delays that are based on the assumption that the target is at a particular angle. The target signal should then be enhanced, since (if the time delays are accurate) each hydrophone is receiving the same signal value from the given target. On the other hand, the noise components will to some extent cancel each other out since they should be incoherent (not synchronized in any way) at the various hydrophones. More

precisely, given the assumption that the noise at each hydrophone is independent identically distributed (iid) from $N(0, \sigma^2)$, the average of n independent noises will be distributed as $N(0, \sigma^2/n)$, where n = the number of hydrophones. Thus, the signal-to-noise ratio is increased by a factor equal to the number of hydrophones. There are sophisticated techniques which attempt to also minimize the effects of the signals from other targets. Thus, the beam former has identified the direction in which one of the targets lies and has produced an enhanced target signal which can then be analyzed by the univariate techniques. (Of course, the quantities involved aren't exact and so statistical techniques are utilized heavily.)

Naturally, there can be difficulties in this scheme. For example, impulsive noises (such as ice cracking, off-shore drilling platforms, or various animal noises) are very problematic. These high energy noises violate the assumption of iid Gaussian noise, producing what is referred to as a contaminated normal model. In these situations, robust estimation techniques are used. One such technique is the Huber M-estimate, which essentially "Windsorizes" the signal so that the outliers produced by the impulsive noises do not have undue influence. Our simulation includes impulsive noise generation, so will allow testing of procedures dealing with this problem.

There are variants of these schemes which perform such tasks as detection and location of targets based on broadband or transient noises (both of which are generated by the targets). The broadband noise is assumed to be Gaussian, but the successive pulses are highly correlated. Such noise is realized as the flow noise due to the movement of the vessel through the water. Transient noises are (as the name implies) of finite duration and are due to irregularly occurring events on the target such as hatches being opened. Since they are generated on the target and typically have fairly short duration, recognizing this noise gives valuable information about the location of the target. This simulation includes generation of these two types of noises as well.

Chapter 2

The Simulation

2.1 The Functionality of the Simulation

The goal of this project is to produce simulated hydrophone recordings; these recordings take the form of files (one for each hydrophone) which contain real numbers written in ASCII format. These real numbers represent acoustic signals which have been sampled at the rate of 1000 times per second. This sampling rate will allow subsequent programs to detect periodic signals of up to 500 Hertz. Besides the periodic signals representing narrow-band frequency suites from the targets, the recordings will contain transient target noise and multipath signals representing the target signals reflecting off the ocean floor or surface. Additionally, local to each hydrophone, Gaussian white noise and impulsive noise are added to the recording. These local noises are often the dominant portion of the recording.

2.2 The Flow of Data Among the Nodes

During the simulation, the Paragon functions as a true Multiple Instruction, Multiple Data (MIMD) machine with three types of nodes. There is a single manager node, a node for each hydrophone, and a node for each target. During each of the simulated time cycles (representing 1/1000 of a second), separate signal pulses are passed from each target node to all of the hydrophone nodes. Also, the manager node passes synchronization messages to each target node and receives synchronization messages from each hydrophone node. So, in addition to using the Paragon as a MIMD machine, the simulation makes extensive use of the Paragon's message passing capabilities.

The manager node initially solicits and reads the various user inputs concerning the number and locations of the targets, the number and locations of the hydrophones, and the noise generation parameters. The manager node then synchronizes the time cycles. Figure 2.1 diagrams the flow of data which occurs during each time cycle.

2.3 What Each Node Does

The Manager Node solicits and reads the user inputs concerning the numbers and locations of the targets and hydrophones. Between inputs, the manager node also computes a matrix of distances between hydrophones and targets, followed by a matrix of time delays which the hydrophone nodes will use to account for the differing arrival times of the signals from the various targets. Using the time delay information, the manager node then runs the simulation for the necessary number of time cycles as a warm-up period to coordinate the simultaneous arrival at a

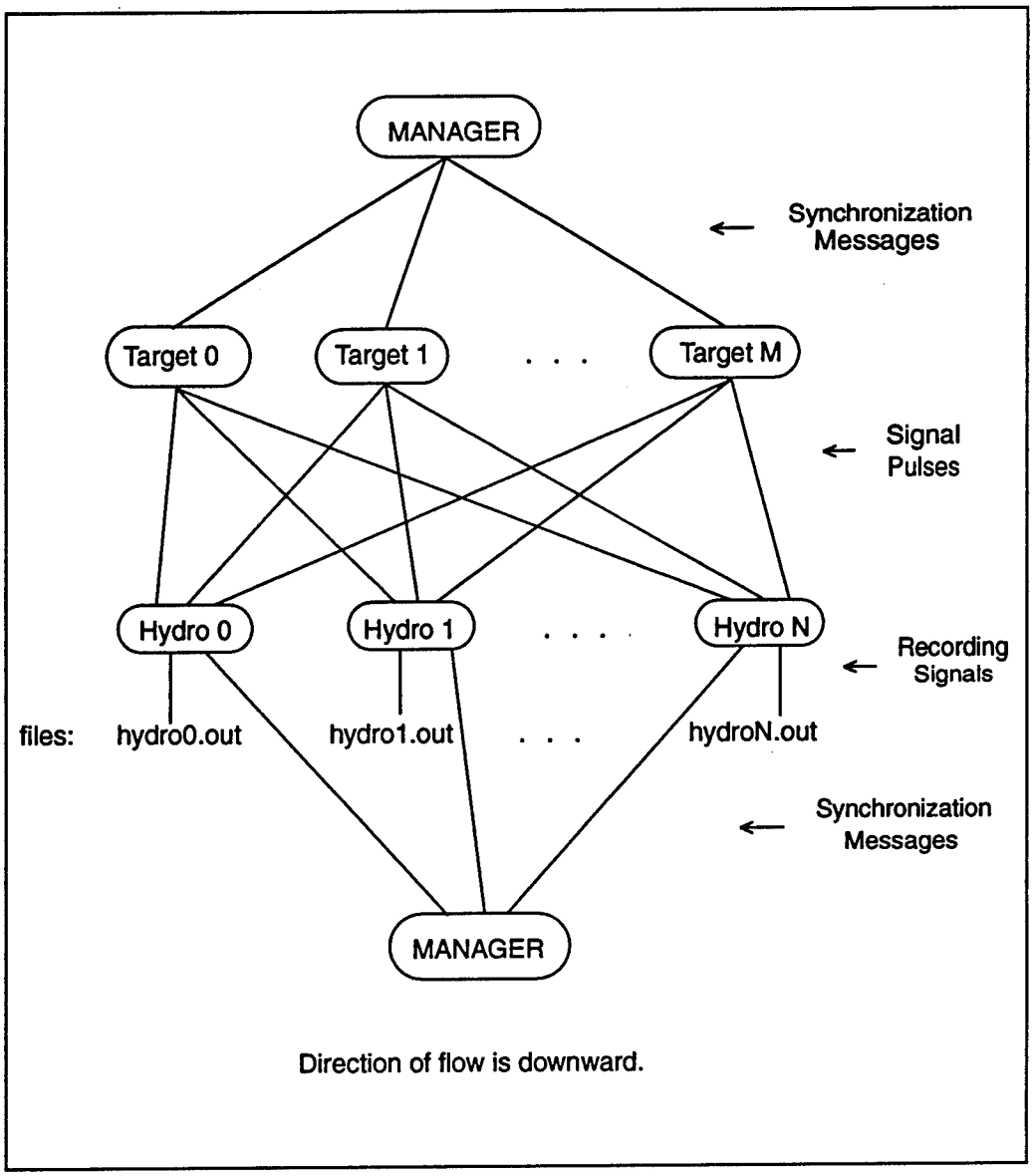


Figure 2.1: Flow of data during each time cycle

given hydrophone of signals that originated at different times. Subsequently, the manager node synchronizes the actual simulation, which runs for the user-specified number of seconds.

The Target Nodes initially receive the user inputs they need and perform some initializations, mainly involving random number generation (which is used for producing the noise components of the signal). The Gaussian random number initialization consists of simply reading from a file an array of precomputed standard normal random numbers. Subsequently, random numbers from a Gaussian distribution with specific mean and standard deviation are produced by multiplying by the standard deviation and adding the mean. During the simulation, a target node generates its narrow-band harmonic frequency suite, broad-band noise, and occasional transient, damped harmonic noise. The combined signal pulse is then sent to each hydrophone node.

The Hydrophones Nodes initially receive the user inputs they need and perform some initializations, again mainly involving random number generation. Each hydrophone node also opens a file for recording and initializes an array for properly temporally assembling the signals, most of which arrive "at the wrong time." During the simulation of the time cycles, the messages containing the signal pulses are placed in the array based on the time delay matrix. As the array is being written to the file, the incoherent Gaussian noise and the occasional impulsive noise are added. In the future, when this project serves as the basis for the virtual CIC project, the array values and added noise will be sent as messages to array processing nodes.

2.4 Interesting Parallel Programming Issues

The challenges in this parallel program lie in two areas. First, the parallel programming and communication among the nodes takes careful planning and testing. This project was developed incrementally. After the user input section was written, the remaining nodes simply received these values from the Manager Node as messages. When this basic communication was functioning correctly, the main loop and the synchronization messages were added, with the nodes still not doing any "useful" work. Next, the features were added and tested one at a time. First, the target nodes generated constant signals and passed those to the hydrophone nodes. Then the targets generated the narrow-band signal suite, which is just a cosine polynomial with frequencies and amplitudes as specified by the user. The remaining features added were white noise at the hydrophone nodes, impulsive noise at the hydrophone nodes, transient noise at the target nodes, and multipath calculation (performed at the hydrophone nodes). Adding the features incrementally with the message passing as a stable foundation made the debugging go fairly smoothly.

The other challenging area was designing the time delay scheme. The time delays reflect the distances between the hydrophone and target pairs in the physical situation being simulated. Two observations guided the time delay algorithm. First, the targets are assumed to be completely independent with regard to the signals they are producing. Hence, for each target, the time delays for that target and all of the hydrophones are calculated without regard to the other targets. This observation greatly shortens the warm-up period. For example, if Target 0 is 2000 meters from Hydrophone 0 while Target 1 is 3000000 meters from Hydrophone 0, there would be an extremely lengthy period of waiting (over 32 minutes) for the signal from Target 1 to get to Hydrophone 0 relative to the wait for the signal from Target 0. By assuming the targets are independent, I can simply assume that for a given time cycle of the simulation, the signal Target 1 generates was

"actually" generated some 32 minutes earlier. Note that since the narrow-band portion of the signal is periodic and the transient noise occurs at random intervals, this assumption of independence of the targets is completely consistent with the physical situation.

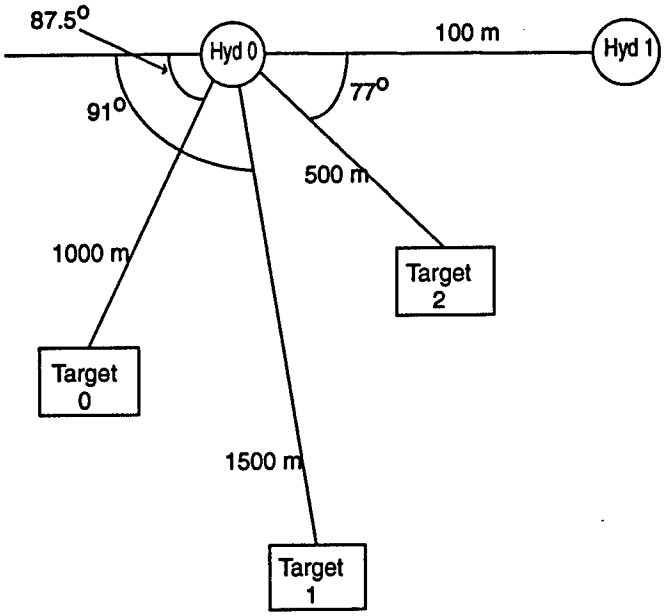
The second observation was simply an extension of the first. The algorithm assumes that the signal pulse generated during a particular time cycle arrives **instantly** at the **furthest** hydrophone from that target. The signal pulse arrives at the closer hydrophones during earlier time cycles. This time manipulation is, of course, possible since this is a simulation, not a recording of an actual event. These signal pulses arriving at "earlier" times is the raison d'etre for the warm-up period. The warm-up period simply runs the simulation for enough time cycles so that all of the pulses sent back from subsequent time cycles are present when the recording of the hydrophone is sent to the file. The source code, of course, completely specifies the algorithm and can be found in the appendix. However, the following example illustrates the entire workings of the time delay algorithm.

An Example to Illustrate the Time Delay Algorithm

This example uses two hydrophones and three targets arranged as illustrated in figure 2.2. There are three parts to this example: first, only Target 0 is considered (see figure 2.3) ; second, Targets 0 and 1 are considered (see figure 2.4); and third, all three targets are considered (see figure 2.5). For each part, the simulation lasts 0.01 seconds (10 time cycles).

Example illustrating how time delays between hydrophones and targets are managed.

This example consists of 2 hydrophones and 3 targets arranged as illustrated below. There are 3 parts to this example: first, only target 0 is considered; second, targets 0 and 1 are considered; and third, all three targets are considered. For each part, the simulation lasts 0.01 seconds (10 time cycles).



By using the Law of Cosines, the speed of sound = 1531 m/sec (in sea water), and 1000 time cycles per second, we compute the following table.

	Distance to Hyd 0	Distance to Hyd 1	Difference in distance	Delay in in secs	Delay in time cycles	Relative delay H 0	Relative delay H 1
Tar 0	1000 m	1009.32	9.32	.0061	6	6	0
Tar 1	1500 m	1501.59	1.59	.0010	1	1	0
Tar 2	500 m	487.34	-12.66	-.0083	-8	0	8

Figure 2.2: Time Delay Example – Physical Layout, Distances, and Time Delays

Part A: How the algorithm handles time delays if Target 0 were the only target. The warm-up period lasts 6 time cycles. Then the simulation lasts 10 time cycles.

In what follows, W0 denotes the pulse sent during warm-up cycle 0, W1 denotes the pulse sent during warm-up cycle 1, etc. A0 denotes the pulse sent during cycle 0 of the actual simulation, A1 denotes the pulse sent during cycle 1 of the actual simulation, etc.

In this part, there is only one target, so each element of the sum array consists of a single pulse from the target.

Only the items enclosed in the rectangles are recorded; the other items are simply discarded. They occur here only to illustrate the algorithm.

Index	Sum Array at Hydrophone 0	Sum Array at Hydrophone 1
-6	W0	
-5	W1	
-4	W2	
-3	W3	
-2	W4	
-1	W5	
0	A0	W0
1	A1	W1
2	A2	W2
3	A3	W3
4	A4	W4
5	A5	W5
6	A6	A0
7	A7	A1
8	A8	A2
9	A9	A3
10		A4
11		A5
12		A6
13		A7
14		A8
15		A9

Note that each pulse arrives 6 time cycles later at Hydrophone 1, as expected.

Figure 2.3: Time Delay Example – Only 1 Target

Part B: How the algorithm handles time delays when there are two targets, Target 0 and Target 1. Note that the actual difference in time between the two targets is ignored. Each target is assumed to be completely independent of the other targets. For each target, time delays are computed to the various hydrophones, ignoring the other targets. In particular, for each target and hydrophone pair, index = cycle - (relative delay). Also note that, in general, the pulses from different targets are different; e.g., A3 from Target 0 is not equal to A3 from Target 1, except by an incredible coincidence.

Again, the warm-up period lasts 6 time cycles, followed by the 10 time cycles of the actual simulation.

Index	Sum Array at Hydrophone 0			Sum Array at Hydrophone 1		
	From Tar 0		From Tar1	From Tar 0		From Tar 1
-6	W0					
-5	W1					
-4	W2					
-3	W3					
-2	W4					
-1	W5	+	W0			
0	A0	+	W1	W0	+	W0
1	A1	+	W2	W1	+	W1
2	A2	+	W3	W2	+	W2
3	A3	+	W4	W3	+	W3
4	A4	+	W5	W4	+	W4
5	A5	+	A0	W5	+	W5
6	A6	+	A1	A0	+	A0
7	A7	+	A2	A1	+	A1
8	A8	+	A3	A2	+	A2
9	A9	+	A4	A3	+	A3
10			A5	A4	+	A4
11			A6	A5	+	A5
12			A7	A6	+	A6
13			A8	A7	+	A7
14			A9	A8	+	A8
15				A9	+	A9

Figure 2.4: Time Delay Example – 2 Targets

Part C: How the algorithm handles time delays for all three targets. The warm-up time for this part lasts 8 time cycles since the largest relative time delay is 8 cycles.

The same notational conventions as in parts A and B still hold.

Index	Sum Array at Hydrophone 0			Sum Array at Hydrophone 1		
	Target 0	Target 1	Target 2	Target 0	Target 1	Target 2
-8						W0
-7						W1
-6	W0					W2
-5	W1					W3
-4	W2					W4
-3	W3					W5
-2	W4					W6
-1	W5 +	W0				W7
0	W6 +	W1 +	W0	W0 +	W0 +	A0
1	W7 +	W2 +	W1	W1 +	W1 +	A1
2	A0 +	W3 +	W2	W2 +	W2 +	A2
3	A1 +	W4 +	W3	W3 +	W3 +	A3
4	A2 +	W5 +	W4	W4 +	W4 +	A4
5	A3 +	W6 +	W5	W5 +	W5 +	A5
6	A4 +	W7 +	W6	W6 +	W6 +	A6
7	A5 +	A0 +	W7	W7 +	W7 +	A7
8	A6 +	A1 +	A0	A0 +	A0 +	A8
9	A7 +	A2 +	A1	A1 +	A1 +	A9
10	A8 +	A3 +	A2	A2 +	A2	
11	A9 +	A4 +	A3	A3 +	A3	
12		A5 +	A4	A4 +	A4	
13		A6 +	A5	A5 +	A5	
14		A7 +	A6	A6 +	A6	
15		A8 +	A7	A7 +	A7	
16		A9 +	A8	A8 +	A8	
17			A9	A9 +	A9	

Figure 2.5: Time Delay Example – All 3 Targets

2.5 Comments on Performance

For small numbers of hydrophones and targets (3 or so of each), the simulation currently runs in real time. With larger numbers of hydrophone and target nodes, the simulation runs slower than real time. I hope this situation improves with some planned improvements to the code and when the operating system is upgraded to utilize both processors on each node.

Chapter 3

Testing and Visualization

3.1 Testing Goals and Techniques

Correctness of a computer program is, of course, always desired; hence, testing should always be a major part of the program development. These maxims are particularly appropriate here. Given this project is the foundation for the larger CIC project and can be used to test new CIC modules, it is essential that this project works correctly. This chapter describes the testing of this project, which provides strong evidence of its correctness. One caveat: there is no input filtering performed; it is the user's responsibility to correctly enter the inputs.

The testing has been accomplished mostly by visualization of the recorded signals using the visualization package AVS. For some simple cases, the Unix utilities, diff and split, were used. For one very small simulation (0.01 seconds), testing was done by inspection of the target outputs, the hydrophone recordings, and the distances between the pairs of hydrophones and targets. This complete inspection is quite tedious and is impractical for larger test cases.

3.2 Simple Test Cases

Simple Harmonic Signal with and without Gaussian Noise

This test case illustrates the effect of white noise. This simple test was run with only one target and one hydrophone. Originally, the target emitted a narrow-band signal formed by the following frequency and amplitude pairs: (10,10) and (20,20). Since the periods (in time cycles) are 100 and 50, respectively, the overall period of the combined harmonic signal is 100, as can be seen in figure 3.1. The simulation was run with no noise. Then the simulation was repeated with Gaussian noise having mean = 0 and SD = 10. The two resulting recordings were overlaid as shown in figure 3.1.

Time Delay Example

Appendix B contains a complete listing of output which tests the algorithm given in the time delay example of the previous chapter. The simulation was modified to produce additional output so the signals produced by the targets can be assembled as diagramed in figures 2.3, 2.4, and 2.5. This target output was assembled and checked against the hydrophone recordings by hand; the tedious arithmetic details are omitted here.

Three Test Cases

Figure 3.2 diagrams three simple test cases. None of these test cases uses local noise; in this way, the only signal is that from the target. So, the only difference in the recordings of Hydrophone 0 and Hydrophone 1 is a shift in time. The first test case simply tests the time delay algorithm in the case where the only target is collinear with the two hydrophones. The 1531 meters is used since the speed of

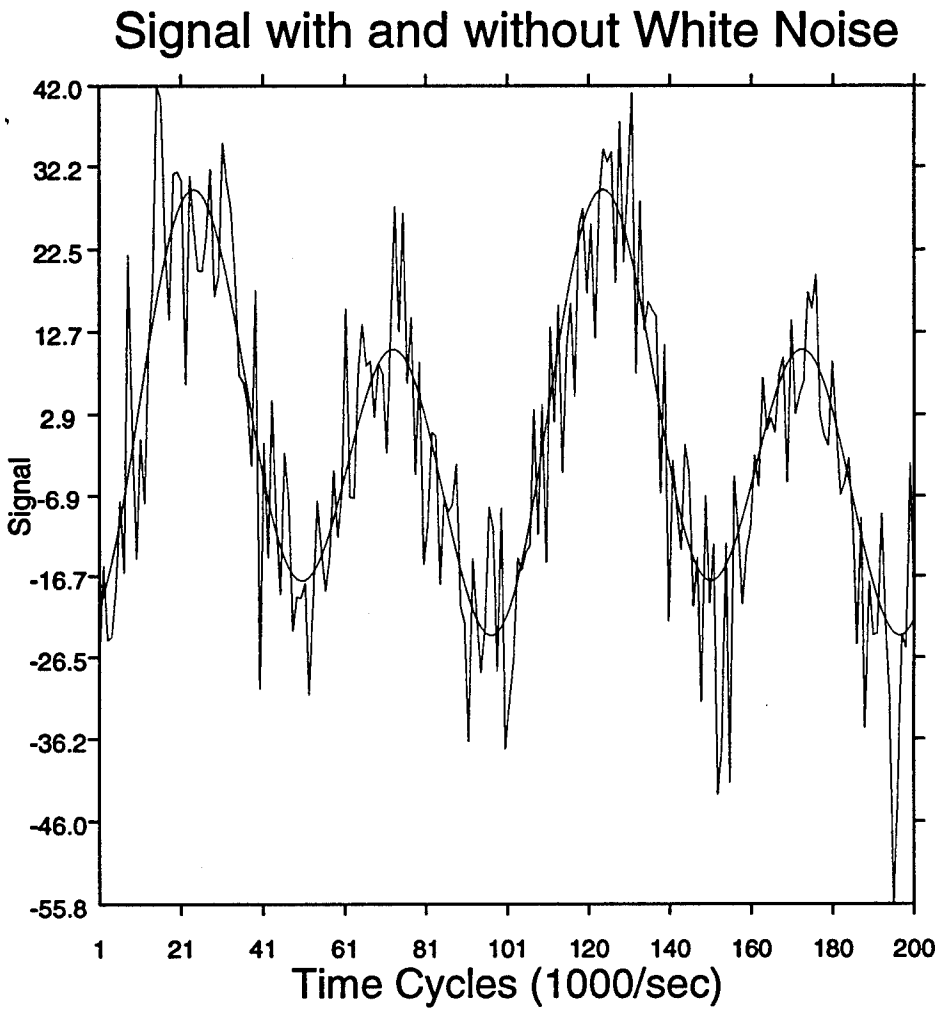


Figure 3.1: Simple Narrow-band Signal

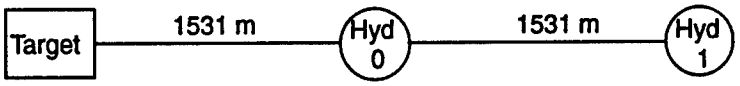
sound in sea water is 1531 meters/second. Having a 1 second (1000 time cycles) time delay is convenient since the Unix utility "split" splits files into files of 1000 lines each.

The second test case is similar, but with two modifications. One is that Hydrophone 1 is closer to the target in this case. The other is the triangular layout to test the distance calculations were correct.

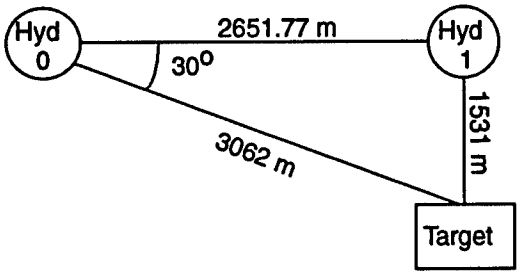
The third test case uses a more arbitrary triangle and is verified graphically. The two graphs (figures 3.3 and 3.4) show both the correctness of the time delay calculations and an occurrence of transient target noise.

Three simple test cases involving 2 hydrophones and 1 target.

1. The recordings at Hydrophone 0 are 1 second (1000 time cycles) ahead of those at Hydrophone 1. This was verified using the Unix utilities "split" and "diff" (with no local noise at the hydrophones).



2. The recordings at Hydrophone 1 are 1 second ahead of those at Hydrophone 0. Again, this was verified using "split" and "diff" with no local noise at the hydrophones.



3. The recordings at Hydrophone 0 are 112 time cycles ahead of those at Hydrophone 1. (The time delay is approximately 0.11175 seconds.) This is illustrated by the following two graphs of the recorded signals at the two hydrophones. These graphs also illustrate the occurence of transient target noise. In these graphs, the target is producing a signal of 20 Hz with amplitude 5. The transient noise is a damped harmonic with initial amplitude of about 20 (it's randomly adjusted) at a frequency of 200 Hz.

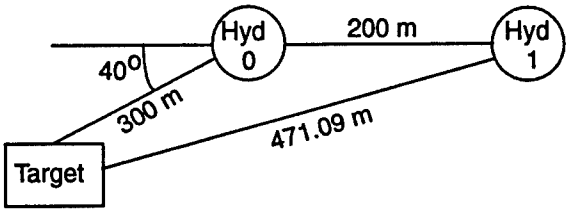


Figure 3.2: Three Test Cases

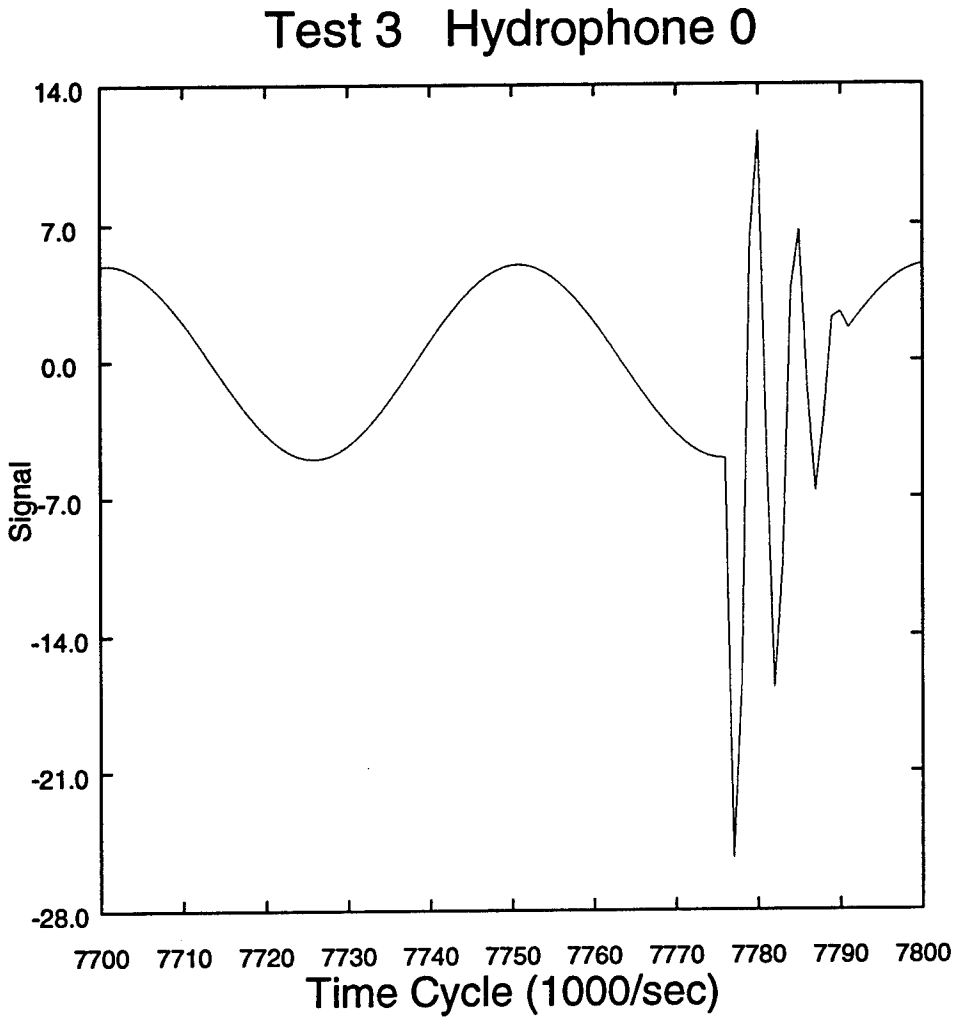


Figure 3.3: Graph Illustrating Test 3

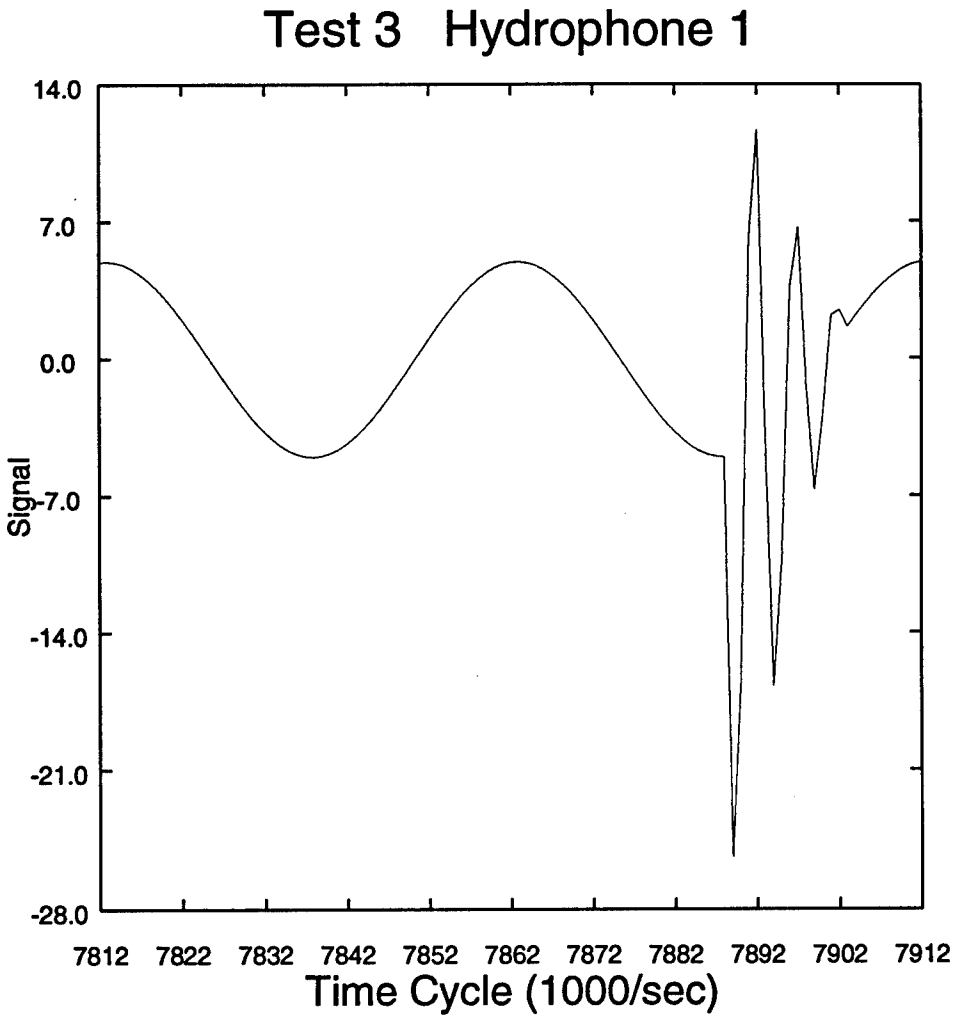


Figure 3.4: Graph Illustrating Test 3

3.3 A Sample Test Case

This section consists of a selection of graphs from one of the test cases that were visualized. The parameters of the simulation are included in Appendix C. This example is typical of the testing performed by use of visualization tools.

Figure 3.5 shows the recording of Hydrophone 0 for the entire 20 second simulation. The large spike is an example of local impulsive noise at the hydrophone. Also note the low small spike between 10000 and 12000 cycles.

Figure 3.6 shows a close-up of the impulsive spike and also illustrates the period signal overlaid with white noise.

Figure 3.7 shows a close-up of the low small spike. While it doesn't appear to be all that impressive, it's about twice as low as any of the surrounding readings. The vertical scale was dominated by the impulsive spike. This low spike is an occurrence of transient target noise as we will see in examining figure 3.9.

Figure 3.8 shows the recording of Hydrophone 1 for the same simulation. Again the large spike is the local impulsive noise. The occurrences of these spikes is different for each hydrophone. (The random number generation was initialized using the node number.)

Figure 3.9 shows the low spike as recorded by Hydrophone 1. Note this spike goes approximately as low as the spike in figure 3.7. (They shouldn't be identical because each hydrophone has independent local white noise.) The time delay is exactly 1000 time cycles, which is the time delay for Target 1 with respect to these two hydrophones. Thus, this is (almost certainly) an example of transient target noise from Target 1.

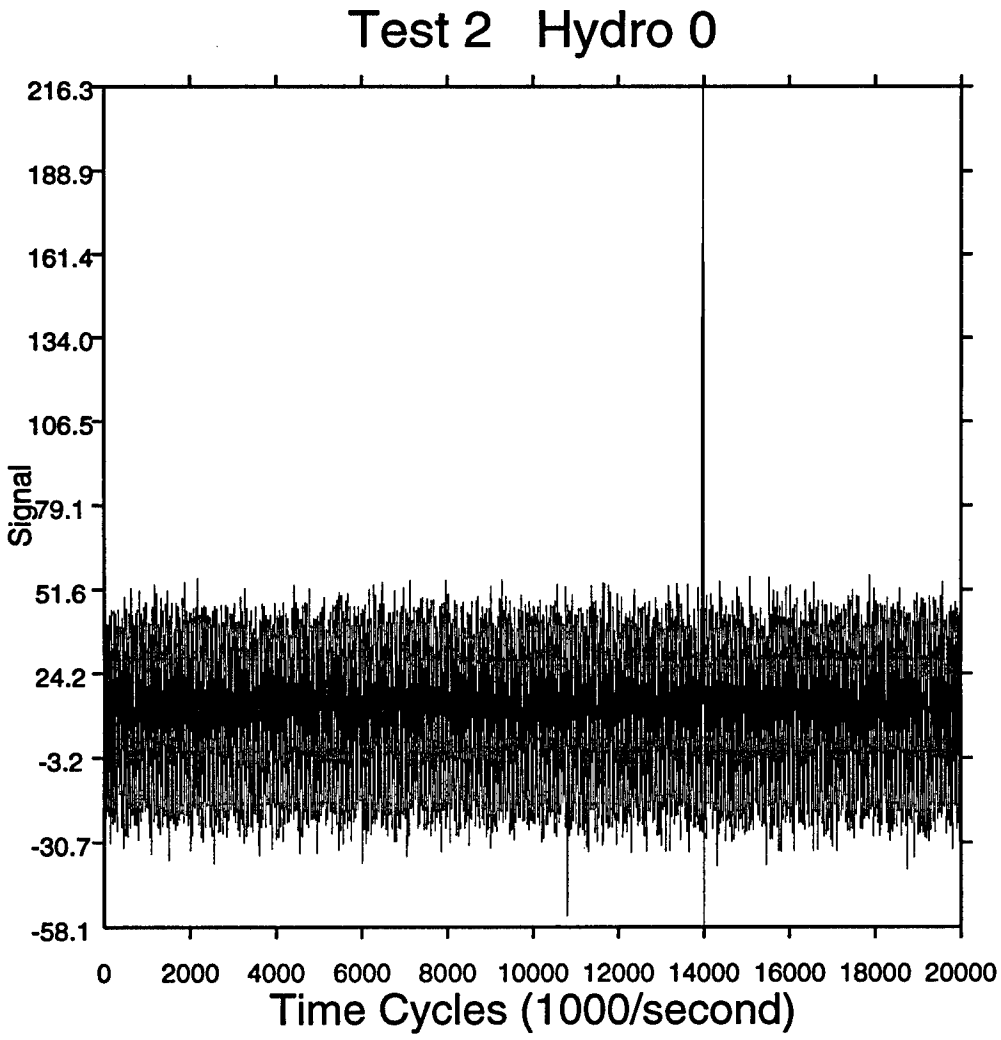


Figure 3.5: Entire Recording at Hydro 0

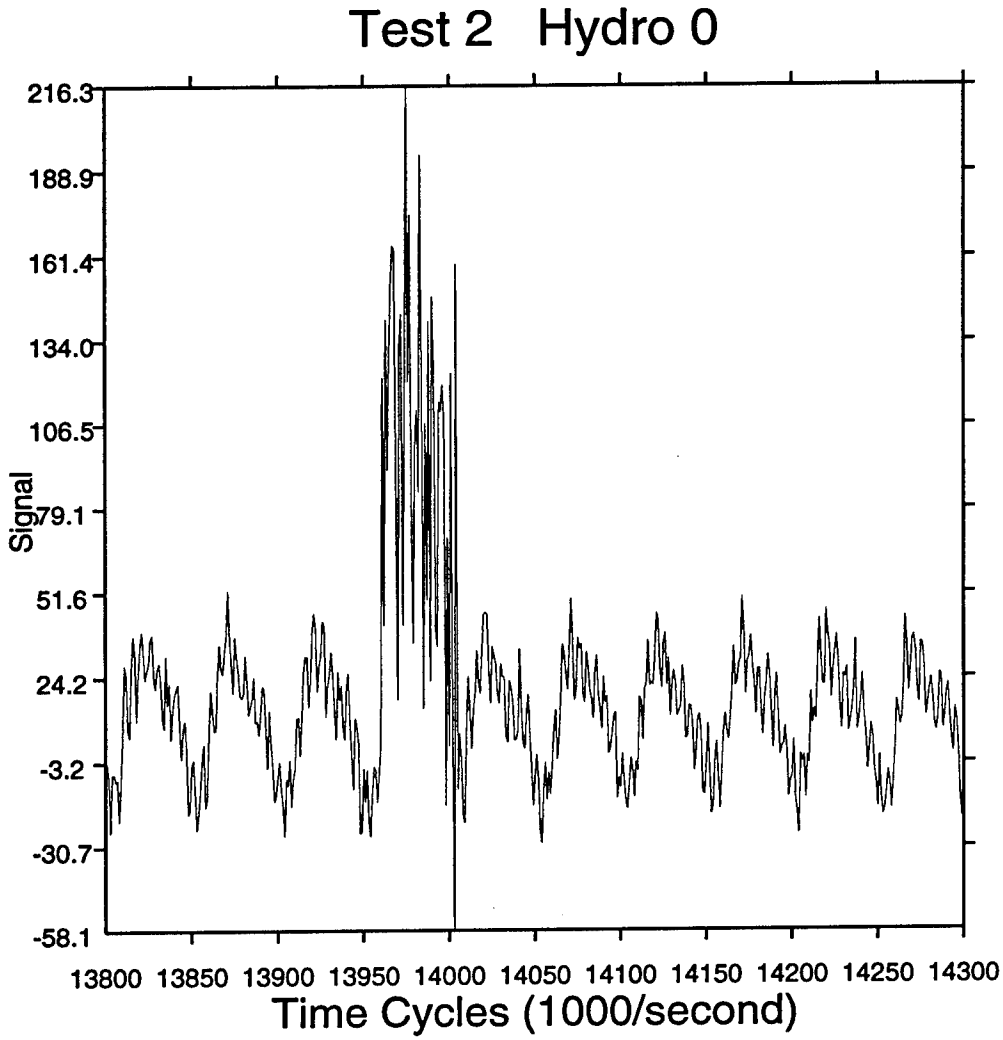


Figure 3.6: Periodic Structure and Impulsive Noise

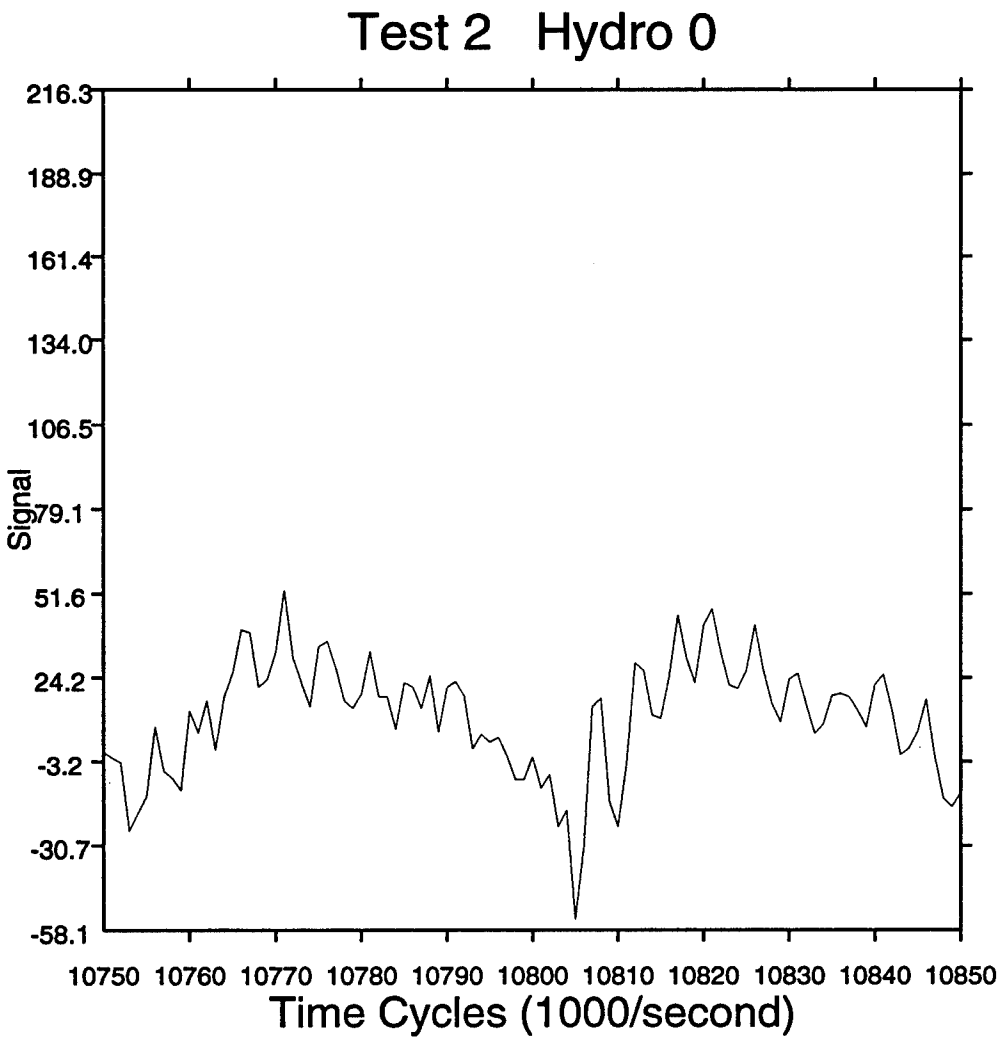


Figure 3.7: Transient Target Noise

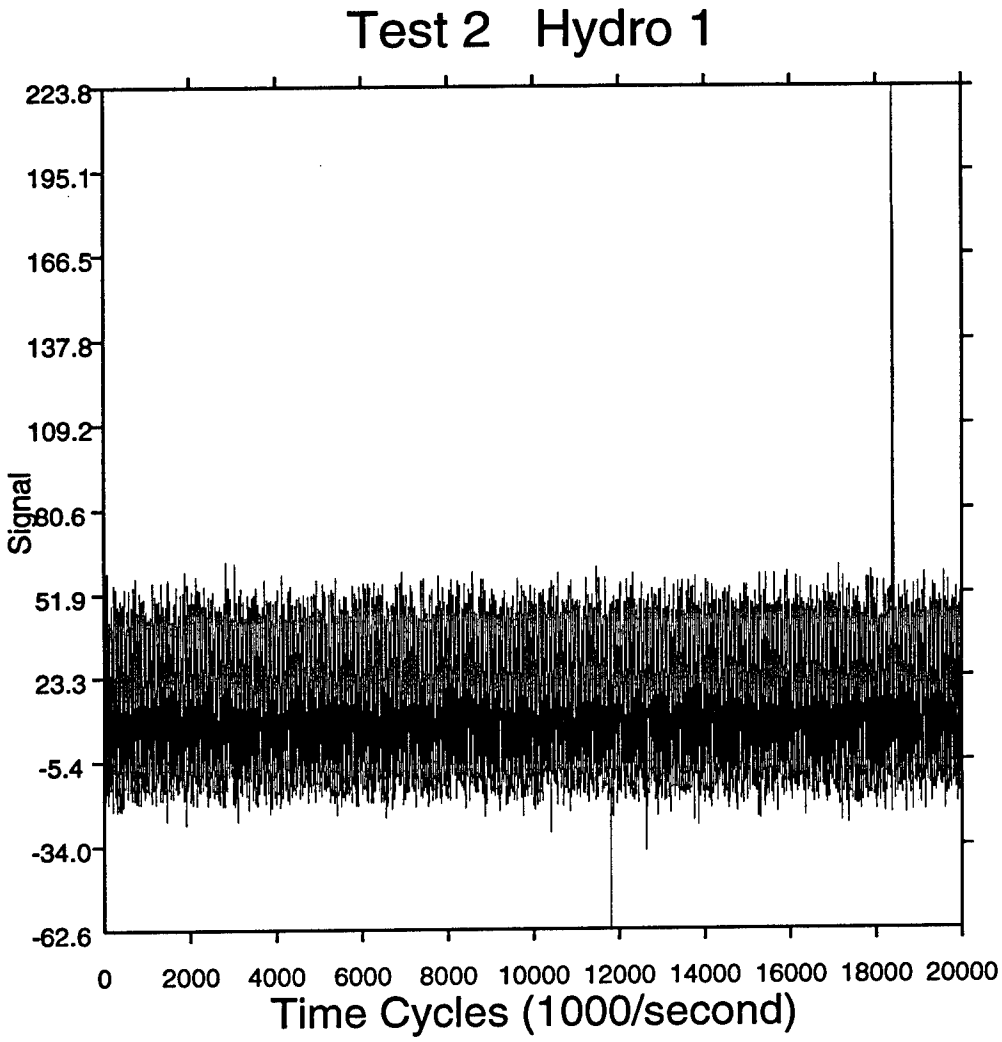


Figure 3.8: Entire Recording at Hydro 1

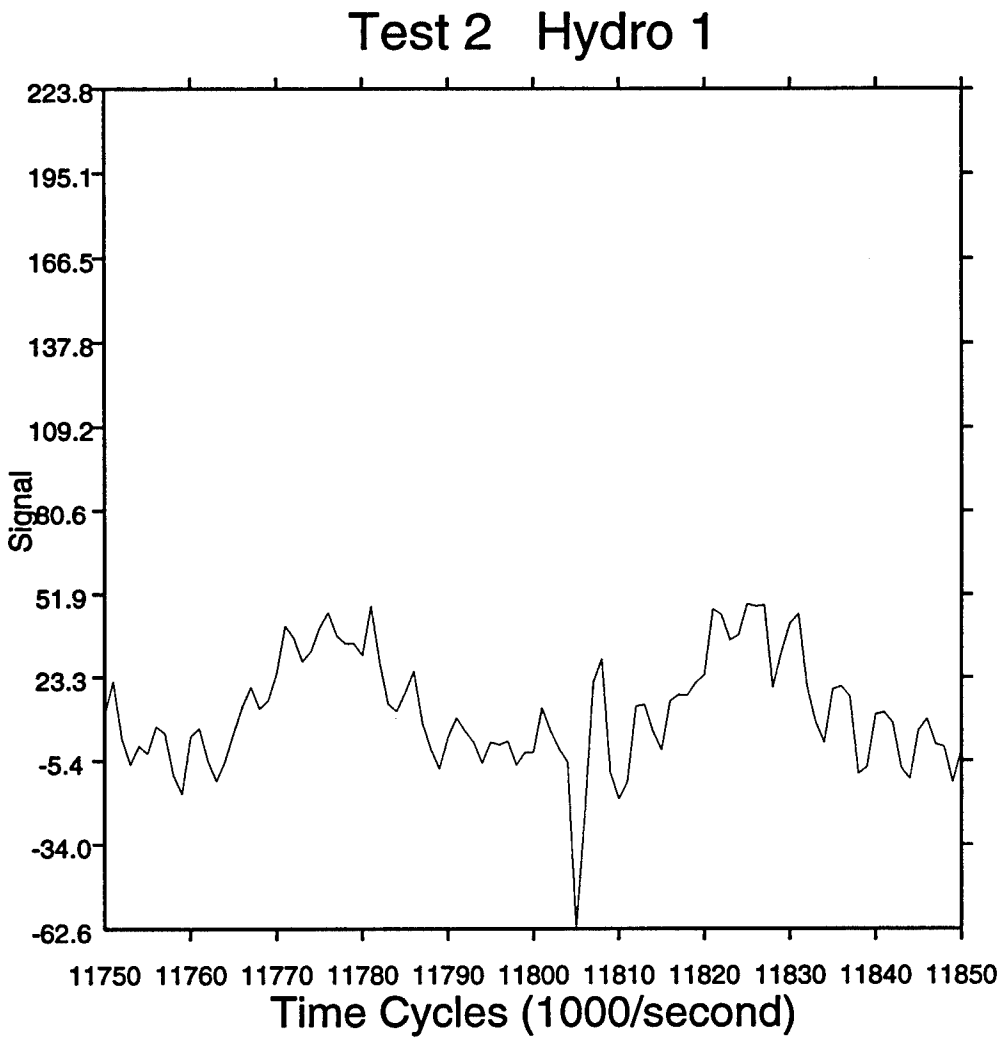


Figure 3.9: Transient Target Noise Revisited

Bibliography

[1] Hall, D. L. (1992). *Mathematical Techniques in Multisensor Data Fusion.* Artech House, Boston.

[2] Oppenheim, A. V. & Schafer, R. W. (1989) *Discrete-Time Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ.

[3] Pillai, S. U. (1989). *Array Signal Processing.* Springer-Verlag, New York.

[4] Stearns, S. D. & David, R. A. (1988). *Signal Processing Algorithms.* Prentice-Hall, Englewood Cliffs, NJ.

[5] Wegman, E. J. & Smith, J. S. (Eds.) (1984). *Statistical Signal Processing.* Marcel-Dekker, New York.

[6] Wegman, E. J., Schwartz, S. C. & Thomas, J. B. (Eds.) (1988) *Topics in NonGaussian Signal Processing.* Springer-Verlag, New York.

Appendix A

The Source Code for the Simulation and Related Files

This appendix contains 6 files: the three node programs written in C, the Makefile used to compile them, the file of constants, and the Unix script file to run the programs on the Paragon.

```
/*
############################
############################
##  MANAGER NODE PROGRAM  ##
############################
############################

File:  /home/cjones/project/signal.c

Use "make" to compile and "sig" in the testsite subdirectory to run.
It's imperative that the programs (signal, hydro, and target) be run
on the correct nodes.  The message passing depends on it.

Note that any changes to the constants MAXHYDRO or MAXSOURCE would require
the corresponding change to be made to sig.

SIGNAL EMULATION PROJECT
PROGRAMMER:  Charles A. Jones
DATE OF LAST REVISION:  July 3, 1994

The purpose of this project is to emulate the signals that would be
recorded by a linear array of hydrophones from various sources, with
noise occurring at the sources, at the hydrophones, and, in future
versions of this project, along multiple paths.


*/

#include "/home/cjones/project/constants.h"
#include <time.h>

void main() {

/*
This program runs on the manager node;  it gets the various inputs
and distributes them to the proper nodes.  Then the manager synchronizes the
time cycles.
```

```
*/

  int i,j,k ;                  /* loop counters */
  int tar, hyd; /* indices for targets and hydros */
  int numHydros;             /* the number of hydrophones */
  float dist[MAXHYDRO]; /* the distance to 0th hydrophone from other hydros */
  double distMat[MAXSOURCE][MAXHYDRO]; /* distances from targets to hydros */
  double distsq;  /* used in law of cosines */

  long cycleDelay[MAXSOURCE][MAXHYDRO]; /* num of cycles for signal to travel
from target to hydro */
  float maxDist[MAXSOURCE];  /* max dist from given target to any hydro */
  long relDelay[MAXSOURCE][MAXHYDRO]; /* relative delay */
  long totalDelay;   /* largest of the relative delays -- length of warm-up */


  int numSource;          /* the number of sources */
  int numFreq[MAXSOURCE];     /* the number of freqencies for each source */
  float freq [MAXSOURCE][MAXFREQ]; /* the freqencies */
  float a [MAXSOURCE][MAXFREQ];  /* the amplitudes -- the cosine coeffs */
  float range [MAXSOURCE]; /* the distance from the source to the 0th hydro */
  float theta [MAXSOURCE];  /* the angle formed by the line of hydros and the
                               line through the source and the 0th hydro */
  float broadMean [MAXSOURCE];  /* the mean of the gaussian forming the
                               broadband portion of the signal */
  float broadSD [MAXSOURCE];  /* the SD of the gaussian for broadband */
  float transAm; /* amplitude of the transient noise at a target */
  float trHowOften; /* average time between transient signals */

  char mulPathAns[40]; /* read answer to multipath prompt */
  int multiPath; /* logical variable --True for multipath included */

  float howLong ;    /* the number of seconds the simulation is to run */
  long numCycles;  /* the total number of cycles = howLong*SAMPLERATE */
  long cycle;  /* loop counter for cycles */

  float noiseMean; /* the mean of the Gaussian noise at each hydro */
```

```
  float noiseSD;   /* the std. dev. of the noise at each hydro */

  float floatDummy; /* these variables are used for message passing */
  int intDummy;
  long longDummy;
  long longSize;
  long floatSize;
  long intSize;

  double startTime; /* stores starting time for computing elapsed time */
  double endTime; /* stores ending time */
  double elapsedTime;    /* elapsed time */

  FILE *fopen(); /* file opening function */
  FILE *fp; /* file pointer  */

/* distance from hydro 0 to hydro 0 is set to 0 */
dist[0] = 0.0;

/* open file for storing results of tests */
if ((fp=fopen("test.results","a"))==NULL) {
   fprintf(stderr, "Cannot open test.results from manager node. \n");
} else {
   fprintf(fp, "\n ############################## \n");
   fprintf(fp, " ###  test results follow   ###");
   fprintf(fp, "\n ############################## \n \n");
}  /* end if */


/* set up the message sizes */
floatSize = sizeof(floatDummy);
intSize = sizeof (intDummy);
longSize = sizeof (longDummy);

/* Welcome message */
printf("\n \n");
printf("Hello, welcome to the signal emulator.\n");
```

```
printf("You will be asked to input several parameters for the emulation.\n");
printf("Thank you in advance for your cooperation.\n\n");


/*
#####################################################################
#####################################################################
The next block of code prompts the user for inputs, reads inputs,
and distributes these inputs.
#####################################################################
#####################################################################
*/
/* Get the user inputs */

/* Start with the length of the simulation */
printf("Please enter the length of time, in seconds, to be simulated.\n");
printf("The maximum allowed is %1d seconds. \n \n", MAXSEC);
scanf ("%f", &howLong);
   numCycles = howLong * SAMPLINGRATE ;
   csend (ANY, &numCycles, longSize, TOALL, 0);
   fprintf(fp, "number of seconds = %1.2f \n", howLong);
   fprintf(fp, "number of time cycles = %1d \n \n", numCycles);


/*
######################
# Hydrophone inputs  #
######################
*/
/* get the number of hydrophones and distribute */
printf("Please enter the number of hydrophones. \n");
printf("The maximum allowable is %1d hydrophones.\n \n", MAXHYDRO);
scanf ("%d", &numHydros);
/* Broadcast this value to all nodes */
csend ( ANY, &numHydros, intSize, TOALL, 0);
fprintf (fp, "number of Hydrophones = %1d \n", numHydros);
```

```
/* get the distances of each hydrophone from hydro 0 */
printf
  ("Please enter the %1d distances, in meters, from the 0th hydrophone.\n",
        numHydros-1);
for (i=1; i<numHydros; i++) {
  scanf("%f", &dist[i]);
  /* Broadcast this value to all nodes */
  csend (ANY, &dist[i], floatSize, TOALL, 0);
  fprintf (fp, "distance from hydro %1d to hydro 0 = %1.2f \n",i,dist[i]);
} ;  /* end for */


/* get the noise mean and sd */
printf
  ("Please enter the mean and SD of the Gaussian noise at the hydros.\n");
scanf ("%f", &noiseMean);
scanf ("%f", &noiseSD );
/* Broadcast these to all active hydrophone nodes */
for (i=0; i< numHydros; i++){
  csend (HYD, &noiseMean, floatSize, i, 0);
  csend (HYD, &noiseSD, floatSize, i, 0);
}
fprintf (fp, "\nGaussian noise at hydros mean = %1.2f \n", noiseMean);
fprintf (fp, "Gaussian noise at hydros SD = %1.2f \n \n",noiseSD);


/* Determine if multipath computations are desired */
printf("Would you like multipath computations done?\n");
scanf ("%s",mulPathAns);
multiPath = (mulPathAns[0] == 'y') || (mulPathAns[0] == 'Y');
for (hyd = 0; hyd < numHydros; hyd++) {
  csend (HYD, &multiPath, intSize, hyd, 0);
}
fprintf (fp, "multipath = %1d \n", multiPath);

/*
```

```
###################
#  Target Inputs  #
###################
*/
/* get the number of targets */
printf ("Please enter the number of targets (sources).\n");
printf ("The maximum allowable is %1d sources.\n",MAXSOURCE);
scanf ("%d", &numSource);
/* Broadcast this value to all nodes */
csend (ANY, &numSource, intSize, TOALL, 0);
fprintf (fp,"Number of targets = %1d \n \n", numSource);

/* get transient noise info */
printf ("Enter the average number of seconds between transient noises\n");
printf ("Enter 0 if transient noise is not desired.\n");
scanf("%f", &trHowOften);
fprintf(fp, "average time between transient noises = %1.2f \n",trHowOften);

/* get the required info for each target */
for (tar=0; tar< numSource; tar++) {
  /* first distribute the transient noise time info */
  csend(TARGET, &trHowOften, floatSize, MAXHYDRO + tar, 0);
  printf ("Please enter the distance in meters from hydrophone 0 to \n");
  printf ("source number %1d .\n",tar);
  scanf ("%f", &range[tar]);
   csend (TARGET, &range[tar], floatSize, MAXHYDRO + tar, 0);

  fprintf (fp, " the range to target %1d = %1.2f \n",tar,range[tar]);

  printf ("Please enter the angle (0-180 degrees) formed by the line of \n");
  printf ("hydrophones and the line from hydrophone 0 and source %1d\n",tar);
  scanf ("%f", &theta[tar]);
   csend (TARGET, &theta[tar], floatSize, MAXHYDRO + tar, 0);

  fprintf (fp, "  the angle for target %1d and the hydros = %1.2f \n",tar,
   theta[tar]);
```

```
printf ("Please enter the mean for the broadband signal for source %1d \n",
        tar);

scanf ("%f",&broadMean[tar]);
csend (TARGET, &broadMean[tar], floatSize, MAXHYDRO + tar, 0);
fprintf
  (fp, "  Broadband mean for target %1d = %1.2f \n",tar, broadMean[tar]);


printf
  ("Please enter the SD for the broadband signal for source %1d \n",tar);

scanf ("%f", &broadSD[tar]);
csend (TARGET, &broadSD[tar], floatSize, MAXHYDRO + tar, 0);
fprintf(fp,"  Broadband SD for target %1d = %1.2f \n",tar,broadSD[tar]);

printf
  ("Please enter the amplitude for the transient noise for target %1d.\n",
     tar);
scanf("%f", &transAm);
csend (TARGET, &transAm, floatSize, MAXHYDRO + tar, 0);
fprintf(fp," Transient amplitude for target %1d = %1.2f \n",tar,transAm);


printf("Please enter the number of freqencies for source %1d \n",tar);
printf("The maximum allowable is %1d \n", MAXFREQ);
scanf ("%d", &numFreq[tar]);
csend (TARGET, &numFreq[tar], intSize, MAXHYDRO + tar, 0);
fprintf
  (fp,"  Number of frequencies at target %1d = %1d \n",tar,numFreq[tar]);
fprintf
  (fp,"  Here are the frequency and amplitude pairs for target %1d: \n",
    tar);
for (j=0; j< numFreq[tar]; j++) {
  printf("Please enter a freqency and amplitude\n");
  scanf ("%f", &freq[tar][j]);
  csend (TARGET, &freq[tar][j], floatSize, MAXHYDRO + tar, 0);
```

```
    scanf ("%f", &a[tar][j]);
    csend (TARGET, &a[tar][j], floatSize, MAXHYDRO + tar, 0);
    fprintf(fp, "%20.2f %20.2f \n", freq[tar][j], a[tar][j]);
  } /* end for j */

  fprintf(fp,"\n");

} /* end for tar */

/*
######################################################################
######################################################################
End of user input section.
######################################################################
######################################################################
*/

/*
######################################################################
######################################################################
Do some calculations of distances
using the law of cosines to determine distances
######################################################################
######################################################################
*/

for (tar=0; tar<numSource; tar++) {
  maxDist[tar] = 0.0;
  for (hyd=0; hyd<numHydros; hyd++) {
    /* Use Law of Cosines to determine distance squared */
    /* Using "+" because interior angle is 180-theta */
    distsq = (range[tar])*(range[tar]) + (dist[hyd])*(dist[hyd])
         + 2.0 * (range[tar]) * (dist[hyd]) * cos(theta[tar] * TORAD);

    /* take square root and store in matrix of distances */
    distMat[tar][hyd] = sqrt(distsq);
```

```
    /* Update max dist if needed */
    if (maxDist[tar] < distMat [tar][hyd]) {
maxDist[tar] = distMat [tar][hyd];
    }  /* end if */
  } /* end for hyd */
} /* end for tar */

/*
#####################################################################
#####################################################################
Use the dist matrix to compute relative time delays for each
TARGET
#####################################################################
#####################################################################
*/

totalDelay = 0;
for (tar = 0; tar < numSource; tar++) {
  for (hyd = 0; hyd < numHydros; hyd++) {

    /* Since the targets are separate, we only need to establish temporal
       coordination relative to each target, not simultaneously for all
       of the targets; use +0.5 to round */
    relDelay[tar][hyd] =
((maxDist[tar]-distMat[tar][hyd])*SAMPLINGRATE)/SPEED + 0.5;
/*  Adjust for multipath delays */
if (multiPath) {
  relDelay[tar][hyd] += MULTI1 + MULTI2;
}

    /* these messages are coded by target number * 100, so they don't get
       mixed-up */
    csend ((HYD+100*tar), &relDelay[tar][hyd], longSize, hyd, 0);

    /* Update total delay if necessary -- total delay is the length of the
       warm-up period that will be required  */
    if (totalDelay < relDelay[tar][hyd]) {
```

```
totalDelay = relDelay[tar][hyd];
    } /*  end if  */
  } /*  end for hyd */
} /* end for tar */

/*  Broadcast totalDelay everywhere --  it's the length of the warm-up */
csend(ANY, &totalDelay, longSize, TOALL, 0);

/*
######################################################################
######################################################################
End of time delay computations.
######################################################################
######################################################################
*/

/*
######################################################################
######################################################################
Manage the warm-up period.
######################################################################
######################################################################
*/

printf ("Warming-up -- will require %1d time cycles \n \n", totalDelay);
fprintf (fp,"Warm up takes %1d time cycles \n",totalDelay);

/* synchronize the warm-up cycles  */
for (cycle=0; cycle < totalDelay; cycle++) {

   /* send the message to start the next cycle */
   for (k=MAXHYDRO; k < (MAXHYDRO + numSource); k++) {
      csend (SYNCH, &intDummy, intSize, k, 0);
   }


   /* wait for all of the hydrophones to complete the cycle */
```

```
   for (j=0; j< numHydros; j++) {
      crecv (RESYNCH, &intDummy, intSize);
   } /* end for j */

} /* end for cycle */

printf ("Warm-up period over -- simulation beginning \n \n");
/*
######################################################################
######################################################################
End of the warm-up period.
######################################################################
######################################################################


######################################################################
######################################################################
Manage the actual simulation.
######################################################################
######################################################################
*/


/* start timing */
startTime = dclock();


/* synchronize the cycles */
for (cycle=0; cycle < numCycles; cycle++) {

   /* send the message to start the next cycle */
   for (k=MAXHYDRO; k < (MAXHYDRO + numSource); k++) {
      csend (SYNCH, &intDummy, intSize, k, 0);
   }


   /* wait for all of the hydrophones to complete the cycle */
```

```
   for (j=0; j< numHydros; j++) {
      crecv (RESYNCH, &intDummy, intSize);
   } /* end for j */

} /* end for cycle */

/*
######################################################################
######################################################################
End of actual simulation.
######################################################################
######################################################################
*/

/* finish timing and report results */
endTime = dclock();

elapsedTime = endTime - startTime ;

printf ("\n \n ");

printf("After completing user inputs and the warm-up, the simulation \n");
printf("took %1.3f seconds. \n \n", elapsedTime);

fprintf(fp, "The simulation took %1.2f seconds. \n \n",elapsedTime);

} /* end of signal */
```

```
/*
##############################
##############################
##   TARGET NODE PROGRAM    ##
##############################
##############################

file:  /home/cjones/project/target.c

Use "make" to compile and "sig" in the testsite subdirectory to run.



PROGRAMMER:  Charles A. Jones
Date of last revision:  June 14, 1994

target.c -- this is the program to run on the target nodes.
This program will produce the signals to be generated by one source
(target) and received by all of the hydrophones.

This program simulates a suite of narrow-band harmonic frequencies,
broad-band flow noise, and transient noise.

*/

#include "/home/cjones/project/constants.h"

void main () {
/*
this program is run on the target nodes.
It accepts the globally broadcast messages, the messages for this particular
target, then computes and broadcasts the signal value for each hydrophone
for each time interval.

*/

long numCycles; /* the number of time cycles */
```

```
long cycle; /* loop counter for time cycles */
long warmUp; /* the number of time cycles in warm-up */
int numHyd; /* the number of hydrophones */
int numTar; /* the number of targets */
float dist[MAXHYDRO]; /* the distances from hydro i to hydro 0 */
int i, j, k,m, n; /* loop counters and other utility ints */
unsigned nn; /* node number */
float t; /* time of current cycle */
float range; /* distance from target to hydro 0 */
float theta; /* angle formed by target and hydro line */
float broadMean; /* mean for broadband signal */
float broadSD; /* standard dev for broadband signal */
int cycleShift; /* number of cycles to shift for time delay */
float broad; /* broadband signal */
float broadArray[BROADMOD];  /* array of precomputed broadband values */
float broadSave[BROADBAND];  /* array used to computed moving average */

int transCount; /* counter used for transient signal  */
int transient; /* logical -- true if transient noise is being added */
float transAm; /* amplitude of transient noise */
float savedTrAm; /* saved amplitude of transient noise */
float transInt; /* expected number of secs between transient noises */
long trThresh; /* threshhold value to compare random with */

int numFreq; /* number of frequencies in narrow band suite */
float freq[MAXFREQ]; /* the freqencies */
float a[MAXFREQ]; /* the corresponding amplitudes */
float phase[MAXFREQ]; /* a phase shift for each frequency */

float randNum[RANDMOD]; /* standard normal random values */
int randIndex; /* index into the st. normal values */

float pulse; /* the signal value */
float shift; /* time shift between current hydro and hydro 0 */

FILE *fopen(), *fp, *fp2;
```

```
long longSize; /* next 6 used for message passing */
long floatSize;
long intSize;

long longDummy;
int intDummy;
float floatDummy;

/* set up message sizes */
floatSize = sizeof(floatDummy);
intSize = sizeof(intDummy);
longSize = sizeof (longDummy);

/* get node number */
n = mynode();
nn = n; /* store as unsigned int */

/* receive some globally broadcast values */
crecv (ANY, &numCycles, longSize);
crecv (ANY, &numHyd, intSize);
dist[0] = 0.0;

for (i=1; i< numHyd; i++) {
   crecv (ANY, &dist[i], floatSize);
}

crecv (ANY, &numTar, intSize);

/* the following block of statements are done for active nodes only */
if (n < (MAXHYDRO + numTar) ) {

  /* first read in some standard normal random numbers */
  if ((fp2=fopen("/home/cjones/project/my.rand","r"))==NULL){
    fprintf(stderr,"Cannot open file of random numbers, node %d \n",n);
  } else {
     for (j=0; j<RANDMOD; j++) {
       fscanf(fp2,"%f", &randNum[j]);
```

```
   }  /* end for j */
}  /* end if else */

/* crank out a random value that depends on the node number */
srandom (nn);
randIndex = random();
randIndex = randIndex % RANDMOD;

 /* receive some user inputs that are just for active targets */
 crecv (TARGET, &transInt, floatSize);
 crecv (TARGET, &range, floatSize);
 crecv (TARGET, &theta, floatSize);
 crecv (TARGET, &broadMean, floatSize);
 crecv (TARGET, &broadSD, floatSize);

/* set-up the broadband noise array */
 broad = 0;
 for (j=0; j < BROADBAND; j++) {
    broadSave[j] = randNum[randIndex];
    randIndex = (randIndex + 1) % RANDMOD;
    broad += broadSave[j];
 }  /* end for j */

 /* precompute broadband values */
 for (i=0; i<BROADMOD; i++) {
    k = i % BROADBAND;
    broad = broad - broadSave[k];
    broadSave[k] = randNum[randIndex];
    randIndex = (randIndex+1) % RANDMOD;
    broad += broadSave[k];
    broadArray[i] = broad * broadSD / BROADBAND + broadMean;
 }  /* end for i */

 /* receive the last of the inputs */
 crecv (TARGET, &savedTrAm, floatSize);
 crecv (TARGET, &numFreq, intSize);
 for (j=0; j < numFreq; j++) {
```

```
      crecv (TARGET, &freq[j], floatSize);

      /* convert the frequency into radians */
      freq[j] = 2*PI*freq[j];
      crecv (TARGET, &a[j], floatSize);
   }  /* end for */

   /* precompute some random phase shifts */
   for (k=0; k<numFreq; k++) {
      phase[k]=randNum[randIndex];
      randIndex = (randIndex+1) % RANDMOD;
   }


/* receive the number of cycles in the warm-up */
crecv (ANY, &warmUp, longSize);

/* set-up transient noise variables */
trThresh = (transInt > 0.0 ) ? TWOto31/(transInt*SAMPLINGRATE) : 0;
transient = (random() < trThresh);
transCount = TRLENGTH;

   /* now do the work */
   /* loop through ALL of the time cycles */
   /* Note that the target nodes do the same job during warm-up as
      during the actual simulation, so the following loop does both */

   for (cycle=0; cycle < (numCycles + warmUp); cycle++) {

      /* convert the cycle into a time in seconds */
      t=(float)cycle/SAMPLINGRATE;

      /* get synchronized */
      crecv (SYNCH, &intDummy, intSize);

      /* build up the narrow band signal (just a trig polynomial) */
      pulse = 0;
```

```
      for (k=0; k< numFreq; k++) {
         pulse = pulse + a[k]*cos((t+phase[k])*freq[k]);
      } /* end for k */

      /* now add in the broadband portion */
      m = cycle % BROADMOD;
      pulse += broadArray[m];

      /* sometimes add in the transient noise */
      if (transient) {
        pulse += transAm*((double)transCount/TRLENGTH)*sin(TRMULT*transCount);
        transient = (--transCount);
      } else if (random() < trThresh) {
        transient = 1;
        transCount = TRLENGTH;
        transAm = savedTrAm * 0.5 * (1.0+fabs((double)randNum[randIndex]));
        randIndex = (randIndex+1) % RANDMOD;
      } /* end if else if */



      /* send out the final value to each hydrophone */
      for (j=0; j < numHyd; j++) {
         csend (TARtoHYD, &pulse, floatSize, j, 0);
      }  /* end for j */

   }  /* end for cycle */

} else {
   /*  this is an unused target node */

   /* receive the number of cycles in the warm-up */
   crecv (ANY, &warmUp, longSize);

   if ((fp=fopen ("target.junk","a")) == NULL) {
      fprintf (stderr, "Can't open target.junk from target node %1d \n", n);
```

```
   } else {
      fprintf (fp, "node: %1d -- target %1d is idle \n", n, n-MAXHYDRO);
   } /* end if */
} /* end if */
}  /* end of target node */
```

```
/*
####################################
####################################
##    HYDROPHONE NODE PROGRAM     ##
####################################
####################################

file: /home/cjones/project/hydro.c

Use "make" to compile and use "sig" in testsite subdirectory to run
See notes at beginning of signal.c.


PROGRAMMER:  Charles A. Jones
Date of last revision:  June 15, 1994

hydro.c --
This is the program that produces the output for the HYDROPHONE nodes.
It receives the globally broadcast messages, the messages for this hydrophone
and then loops through all the time cycles.  During each time cycle,
this node receives signal values from each of the targets, assembles them
accounting for time delays (the targets are in different positions), then adds
them together, adds in some noise, and writes the value to the output file
for this hydrophone.

*/

#include "/home/cjones/project/constants.h"



void main () {
/*
this function is run on the hydrophone nodes.
*/

long numCycles;  /* number of time cycles  */
```

```
long cycle; /* loop counter for time cycles */
long warmUp; /* number of cycles in warm-up period */
long offset[MAXSOURCE]; /* cycle offsets for the various targets */
int numHyd; /* number of hydrophones  */
int numTar; /* number of targets */
float dist[MAXHYDRO]; /* distances from hydro i to hydro 0 */
int i, j, k, n; /* loop counters & utility integers */
int whichTar; /* target from which last pulse received */
int tar; /* loop counter for targets */
unsigned nodeNum; /* node number */
char filename[STRLENGTH]; /* filename for output */
FILE *fopen(), *fp, *fp2; /* built-in file open fun,2 file ptrs */

int multiPath; /* logical -- do multipath computations? */

float randNum[RANDMOD]; /* previously produced st. normal values */
int randIndex; /* index into random values */
int transIndex; /* index used for transient noise  */

float noiseMean; /* mean for Gaussian (white) noise */
float noiseSD; /* standard dev for Gaussian noise */

/* static declarations gaurantee initialization to 0 */
static float sum[MAXCYCLE]; /* sum of signals received from targets */
static long count[MAXCYCLE];
long index;
float pulse; /* recorded signal pulse  */

long floatSize; /* these 6 are used for message passing */
long longSize;
long intSize;
long longDummy;
int intDummy;
float floatDummy;


/* set-up size of various message types */
```

```
floatSize = sizeof(floatDummy);
intSize = sizeof(intDummy);
longSize = sizeof(longDummy);

/* read in some standard normal numbers */
if ((fp2=fopen("/home/cjones/project/my.rand", "r")) == NULL) {
  fprintf (stderr,
  "Cannot open file of random numbers from node %d \n",n);
} else {
  for (j=0; j<RANDMOD; j++) {
     fscanf(fp2,"%f", &randNum[j]);
  }  /* end for j */
}  /* end if else */


/* next receive the user inputs */
crecv (ANY, &numCycles, longSize);
crecv (ANY, &numHyd, intSize);
dist[0] = 0.0;
for (i=1; i< numHyd; i++) {
   crecv (ANY, &dist[i], floatSize);
}

/* get node number */
n = mynode();
nodeNum = n; /* store as unsigned int */


if (n < numHyd) {
   crecv (HYD, &noiseMean, floatSize);
   crecv (HYD, &noiseSD, floatSize);
   crecv (HYD, &multiPath, intSize);
}  /* end if */

crecv (ANY, &numTar, intSize);

if (n < numHyd) {
```

```
/* This is an active hydrophone node */

/* first receive the offsets (measured in time cycles) for this
hydrophone and all of the targets. */
for (tar=0; tar < numTar; tar++) {
  crecv ((HYD + 100*tar), &offset[tar], longSize);
}

/* receive the length of the warm-up period (in time cycles)
   it's the max of the offsets for all targets and all hydros */
crecv(ANY, &warmUp, longSize);

/* open file for output -- node number embedded in file name */
sprintf (filename, "hydro%1d.out", n);
if ((fp=fopen(filename,"w")) == NULL) {
   fprintf(stderr, "Can't open file %s \n", filename);
} else {

   /* crank out a random value that will depend on the hydrophone */
   /* so the different hydrophones exhibit different randomness   */
   srandom (nodeNum);
   randIndex = random();

   /* this is different for each hydrophone */
   transIndex = random();

   /* run through the warm-up period */
   /* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX */
   for (cycle=0; cycle<warmUp; cycle++) {

      for (tar = 0; tar < numTar; tar++) {
         crecv (TARtoHYD, &pulse, floatSize);
         whichTar = infonode() - MAXHYDRO;
         index = cycle - offset[whichTar];
         if ((index >= 0) && (index<numCycles)) {
           sum[index] = sum[index] + pulse;
           count[index] = count[index] + 1;
```

```
            }  /* end if */

            /* HANDLE MULTIPATH CALCULATIONS HERE */
            if (multiPath) {
index += MULTI1;
if ((index >= 0) && (index < numCycles)) {
   sum[index] = sum[index] + pulse * MPRF1;
}  /* end if */
index += MULTI2;
if ((index >= 0) && (index < numCycles)) {
   sum[index] = sum[index] + pulse * MPRF2;
}  /* end if */
    }  /* end if multiPath */

         }  /* end for tar */

         /* Inform MANAGER NODE all messages for this cycle were received */
         csend (RESYNCH, &intDummy, intSize, MAIN, 0);

      } /* end for cycle = warmUp */
      /* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX */



      /* Loop through all the time cycles */
      /* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX */
      /* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX */
      for (cycle=0; cycle < numCycles; cycle++) {

 randIndex = (randIndex + 1) % RANDMOD;

         for (tar=0; tar< numTar; tar++) {
            crecv (TARtoHYD, &pulse, floatSize);
            whichTar = infonode() - MAXHYDRO;
            index = warmUp + cycle - offset[whichTar];
/* it shouldn't be possible for index to be less than 0 */
/* delete after testing XXXXXXXXXXXXXXXXX */
```

```
if (index<0) {
fprintf (stderr, "index less than 0  ERROR CONDITION \n");
}

            if (index < numCycles) {
              sum[index] = sum[index] + pulse;
              count[index] = count[index] + 1;
            } /* end if */

            /* HANDLE MULTIPATH CALCULATIONS HERE */
            if (multiPath) {
index += MULTI1;
if ((index >= 0) && (index < numCycles)) {
   sum[index] = sum[index] + pulse * MPRF1;
} /* end if */
index += MULTI2;
if ((index >= 0) && (index < numCycles)) {
   sum[index] = sum[index] + pulse * MPRF2;
} /* end if */
    } /* end if multiPath */

         } /* end for tar */

         /* Inform MANAGER NODE all messages for this cycle were received */
         csend (RESYNCH, &intDummy, intSize, MAIN, 0);


         /*  add noise here */
         sum[cycle] += noiseMean + noiseSD * randNum [randIndex];

         /* sometimes add transcient noise */
         if (((cycle+transIndex)%TRANSMOD) < TRANSLIM) {
   sum[cycle] +=
      TRANSMF*noiseMean+TRANSSDF*noiseSD*randNum[RANDMOD-(randIndex+1)];
transIndex += 5;  /* shorten the duration of the trans noise */
 }
```

```
/* debugging XXXXXXXX */
if (count[cycle] != numTar) {
  fprintf(stderr, "count was %d expected count was %d when cycle = %d \n\n",
          count[cycle], numTar, cycle);
}


         fprintf (fp, "%20.4f\n", sum[cycle]);

      /* XXXXXXXXXXXXXXXXXXXXXXXXXXXXX */
      /* XXXXXXXXXXXXXXXXXXXXXXXXXXXXX */
      }  /* end for cycle */


   }  /* end of inner if then else */




} else {
   /* this node is an unused hydrophone node */

   /*  need to receive this one last message */
   crecv(ANY, &warmUp, longSize);

   if ((fp=fopen("hydros.junk","a")) == NULL) {
      fprintf(stderr, "Can't open file hydros.junk from node %1d.\n", n);
   } else {
      fprintf (fp, "hydro %1d is idle \n", n);
   }
} /* end if */

}  /* end of hydro */
```

```
################
# constants.h  #
################


#define MULTI1 250
#define MULTI2 150
#define MPRF1 0.25
#define MPRF2 0.2
#define TRANFREQ 200.0
#define TRANSMOD 19473
#define TRANSLIM 264
#define TRANSMF 10.0
#define TRANSSDF 10.0
#define RANDMOD 3449
#define TOALL -1
#define ANY 1
#define HYD 2
#define TARGET 3
#define TARtoHYD 4
#define HYDtoTAR 5
#define HYDtoAP 7
#define SYNCH 10
#define RESYNCH 11
#define STRLENGTH 80
#define MAXHYDRO 30
#define MAXSOURCE 15
#define MAIN (MAXHYDRO + MAXSOURCE)
#define MAXFREQ  20
#define SAMPLINGRATE 1000
#define MAXSEC 300
#define MAXCYCLE (SAMPLINGRATE*MAXSEC)
#define BROADBAND 6
#define BROADMOD 5557
#define SPEED 1531.0
#define PI 3.141592654
#define MAXHZ 500.0
```

```
#define MINHZ 0.0
#define TORAD (PI/180.0)
#define TWOto31 2147483648.0
#define TRLENGTH 15
#define TRTHRESH 1000000
#define TRMULT (0.4*PI)

#include <stdio.h>
#include <nx.h>
#include <math.h>
```

```
#################################################################
# Makefile for compiling the programs hydro, target, signal     #
# with the command "make" or "pmake".                           #
#################################################################

hydro signal target:  hydro.o signal.o target.o Makefile
icc -nx -o hydro -O2 hydro.o
icc -nx -o signal -O2 signal.o
icc -nx -o target -O2 target.o


hydro.o:  hydro.c constants.h
icc -c -O2 hydro.c

target.o:  target.c constants.h
icc -c -O2 target.c

signal.o:  signal.c constants.h
icc -c -O2 signal.c
```

```
#################################################################
# The Unix script "sig".                                        #
# Used to run the simulation.                                   #
# Note that the message passing requires these 3 programs to be #
# run on exactly the nodes specified here.                      #
#################################################################

This file has been modified for presentation;
it should consist of a single line.


/home/cjones/project/signal -sz 46 -on 45 \;
/home/cjones/project/hydro -on 0..29 \;
/home/cjones/project/target -on 30..44
```

Appendix B

Verbose Output to Illustrate the Time Delay Example

This appendix contains first a file which describes the user inputs and time delay calculations. This is followed by a file which contains the pulses sent out by each target as well as the hydrophone recordings. These target pulses were assembled by hand as described in the time delay example and they agreed with the hydrophone recordings.

```
##########################################
###  test results for part (a) follow   ###
##########################################

number of seconds = 0.01
number of time cycles = 10

number of Hydrophones = 2
distance from hydro 1 to hydro 0 = 100.00

Gaussian noise at hydros mean = 0.00
Gaussian noise at hydros SD = 0.00

multipath = 0
Number of targets = 1

average time between transient noises = 6.00
```

```
 the range to target 0 = 1000.00
  the angle for target 0 and the hydros = 87.50
  Broadband mean for target 0 = 3.00
  Broadband SD for target 0 = 4.00
 Transient amplitude for target 0 = 5.00
  Number of frequencies at target 0 = 1
  Here are the frequency and amplitude pairs for target 0:
              60.00                  6.00


 Distance Matrix follows
    1000.000    1009.319

 Time Delay Matrix
         6         0
Warm up takes 6 time cycles
The simulation took 0.06 seconds.


 ###########################################
 ###  test results for part (b) follow   ###
 ###########################################

number of seconds = 0.01
number of time cycles = 10

number of Hydrophones = 2
distance from hydro 1 to hydro 0 = 100.00

Gaussian noise at hydros mean = 0.00
Gaussian noise at hydros SD = 0.00

multipath = 0
```

```
Number of targets = 2

average time between transient noises = 8.00
 the range to target 0 = 1000.00
  the angle for target 0 and the hydros = 87.50
  Broadband mean for target 0 = 2.00
  Broadband SD for target 0 = 3.00
 Transient amplitude for target 0 = 4.00
  Number of frequencies at target 0 = 1
  Here are the frequency and amplitude pairs for target 0:
               60.00                  6.00

 the range to target 1 = 1500.00
  the angle for target 1 and the hydros = 91.00
  Broadband mean for target 1 = 6.00
  Broadband SD for target 1 = 7.00
 Transient amplitude for target 1 = 8.00
  Number of frequencies at target 1 = 1
  Here are the frequency and amplitude pairs for target 1:
               40.00                  4.00


 Distance Matrix follows
    1000.000    1009.319
    1500.000    1501.587

 Time Delay Matrix
         6          0
         1          0
Warm up takes 6 time cycles
The simulation took 0.07 seconds.
```

```
 ##########################################
 ###  test results for part (c) follow   ###
 ##########################################

number of seconds = 0.01
number of time cycles = 10

number of Hydrophones = 2
distance from hydro 1 to hydro 0 = 100.00

Gaussian noise at hydros mean = 0.00
Gaussian noise at hydros SD = 0.00

multipath = 0
Number of targets = 3

average time between transient noises = 8.00
 the range to target 0 = 1000.00
  the angle for target 0 and the hydros = 87.50
  Broadband mean for target 0 = 2.00
  Broadband SD for target 0 = 3.00
 Transient amplitude for target 0 = 4.00
  Number of frequencies at target 0 = 1
  Here are the frequency and amplitude pairs for target 0:
              60.00                  6.00

 the range to target 1 = 1500.00
  the angle for target 1 and the hydros = 91.00
  Broadband mean for target 1 = 5.00
  Broadband SD for target 1 = 6.00
 Transient amplitude for target 1 = 7.00
  Number of frequencies at target 1 = 1
  Here are the frequency and amplitude pairs for target 1:
```

```
              40.00                  4.00

 the range to target 2 = 500.00
  the angle for target 2 and the hydros = 103.00
  Broadband mean for target 2 = 8.00
  Broadband SD for target 2 = 9.00
 Transient amplitude for target 2 = 1.00
  Number of frequencies at target 2 = 1
  Here are the frequency and amplitude pairs for target 2:
              20.00                  7.00


 Distance Matrix follows
    1000.000    1009.319
    1500.000    1501.587
     500.000     487.345

 Time Delay Matrix
         6          0
         1          0
         0          8
Warm up takes 8 time cycles
The simulation took 0.04 seconds.


::::::::::::::
Tars.a.out
::::::::::::::
tar = 0    cycle = 0    pulse = 0.841
tar = 0    cycle = 1    pulse = -1.002
tar = 0    cycle = 2    pulse = -2.882
tar = 0    cycle = 3    pulse = -4.056
tar = 0    cycle = 4    pulse = -3.170
tar = 0    cycle = 5    pulse = -2.782
```

```
tar = 0    cycle = 6    pulse = -0.130
tar = 0    cycle = 7    pulse = 2.767
tar = 0    cycle = 8    pulse = 4.241
tar = 0    cycle = 9    pulse = 6.469
tar = 0    cycle = 10    pulse = 7.543
tar = 0    cycle = 11    pulse = 9.132
tar = 0    cycle = 12    pulse = 8.217
tar = 0    cycle = 13    pulse = 6.507
tar = 0    cycle = 14    pulse = 6.087
tar = 0    cycle = 15    pulse = 2.030
::::::::::::::
Tars.b.out
::::::::::::::
tar = 0    cycle = 0    pulse = 0.008
tar = 0    cycle = 1    pulse = -1.883
tar = 0    cycle = 2    pulse = -3.678
tar = 0    cycle = 3    pulse = -4.765
tar = 0    cycle = 4    pulse = -4.101
tar = 0    cycle = 5    pulse = -3.603
tar = 0    cycle = 6    pulse = -1.230
tar = 0    cycle = 7    pulse = 1.452
tar = 0    cycle = 8    pulse = 3.119
tar = 0    cycle = 9    pulse = 5.324
tar = 0    cycle = 10    pulse = 6.563
tar = 0    cycle = 11    pulse = 8.025
tar = 0    cycle = 12    pulse = 7.410
tar = 0    cycle = 13    pulse = 5.988
tar = 0    cycle = 14    pulse = 5.342
tar = 0    cycle = 15    pulse = 1.825
tar = 1    cycle = 0    pulse = 4.933
tar = 1    cycle = 1    pulse = 3.439
tar = 1    cycle = 2    pulse = 2.573
tar = 1    cycle = 3    pulse = 3.586
```

```
tar = 1    cycle = 4    pulse = 2.017
tar = 1    cycle = 5    pulse = 3.755
tar = 1    cycle = 6    pulse = 3.045
tar = 1    cycle = 7    pulse = 3.596
tar = 1    cycle = 8    pulse = 3.371
tar = 1    cycle = 9    pulse = -1.080
tar = 1    cycle = 10    pulse = 2.021
tar = 1    cycle = 11    pulse = 9.203
tar = 1    cycle = 12    pulse = 12.605
tar = 1    cycle = 13    pulse = 11.050
tar = 1    cycle = 14    pulse = 7.461
tar = 1    cycle = 15    pulse = 8.665
:::::::::::::::
Tars.c.out
:::::::::::::::
tar = 1    cycle = 0    pulse = 4.058
tar = 1    cycle = 1    pulse = 2.637
tar = 1    cycle = 2    pulse = 1.764
tar = 1    cycle = 3    pulse = 2.521
tar = 1    cycle = 4    pulse = 1.090
tar = 1    cycle = 5    pulse = 2.524
tar = 1    cycle = 6    pulse = 1.896
tar = 1    cycle = 7    pulse = 2.384
tar = 1    cycle = 8    pulse = 2.241
tar = 1    cycle = 9    pulse = -1.594
tar = 1    cycle = 10    pulse = 1.216
tar = 1    cycle = 11    pulse = 7.615
tar = 1    cycle = 12    pulse = 10.697
tar = 1    cycle = 13    pulse = 9.427
tar = 1    cycle = 14    pulse = 6.421
tar = 1    cycle = 15    pulse = 7.606
tar = 1    cycle = 16    pulse = 10.929
tar = 1    cycle = 17    pulse = 12.531
```

```
tar = 0    cycle = 0    pulse = 0.008
tar = 0    cycle = 1    pulse = -1.883
tar = 0    cycle = 2    pulse = -3.678
tar = 0    cycle = 3    pulse = -4.765
tar = 0    cycle = 4    pulse = -4.101
tar = 0    cycle = 5    pulse = -3.603
tar = 0    cycle = 6    pulse = -1.230
tar = 0    cycle = 7    pulse = 1.452
tar = 0    cycle = 8    pulse = 3.119
tar = 0    cycle = 9    pulse = 5.324
tar = 0    cycle = 10    pulse = 6.563
tar = 0    cycle = 11    pulse = 8.025
tar = 0    cycle = 12    pulse = 7.410
tar = 0    cycle = 13    pulse = 5.988
tar = 0    cycle = 14    pulse = 5.342
tar = 0    cycle = 15    pulse = 1.825
tar = 0    cycle = 16    pulse = -0.143
tar = 0    cycle = 17    pulse = -1.720
tar = 2    cycle = 0    pulse = 4.039
tar = 2    cycle = 1    pulse = 6.632
tar = 2    cycle = 2    pulse = 4.669
tar = 2    cycle = 3    pulse = 4.312
tar = 2    cycle = 4    pulse = 4.081
tar = 2    cycle = 5    pulse = 4.199
tar = 2    cycle = 6    pulse = 4.093
tar = 2    cycle = 7    pulse = 3.021
tar = 2    cycle = 8    pulse = 1.268
tar = 2    cycle = 9    pulse = 2.455
tar = 2    cycle = 10    pulse = 5.934
tar = 2    cycle = 11    pulse = 8.507
tar = 2    cycle = 12    pulse = 9.827
tar = 2    cycle = 13    pulse = 11.661
tar = 2    cycle = 14    pulse = 16.042
```

```
tar = 2    cycle = 15    pulse = 15.045
tar = 2    cycle = 16    pulse = 14.335
tar = 2    cycle = 17    pulse = 14.510
::::::::::::::
hyd0.a.out
::::::::::::::
             -0.1301
              2.7669
              4.2413
              6.4687
              7.5434
              9.1317
              8.2172
              6.5069
              6.0869
              2.0299
::::::::::::::
hyd0.b.out
::::::::::::::
              2.2097
              4.0246
              6.7046
              7.3408
             10.3178
             11.0707
             11.0058
              9.3582
              4.2619
              3.8456
::::::::::::::
hyd0.c.out
::::::::::::::
              5.4465
```

```
                9.8481
               10.3086
               10.7259
               13.1682
               14.1208
               13.8863
               11.2499
                5.0156
                5.4967
::::::::::::::
hyd1.a.out
::::::::::::::
                0.8409
               -1.0016
               -2.8823
               -4.0559
               -3.1703
               -2.7818
               -0.1301
                2.7669
                4.2413
                6.4687
::::::::::::::
hyd1.b.out
::::::::::::::
                4.9405
                1.5567
               -1.1052
               -1.1791
               -2.0843
                0.1517
                1.8159
                5.0477
```

```
                6.4892
                4.2434
::::::::::::::
hyd1.c.out
::::::::::::::
                5.3339
                3.2102
                4.0202
                6.2620
                6.8151
               10.5825
               16.7090
               18.8802
               19.6947
               18.2399
```

Appendix C

Parameters for Sample Test Case

This appendix contains a file which describes the user inputs for the sample test case of section 3.3. This file is produced by the simulation.

```
 ##############################
 ###  test results follow   ###
 ##############################

number of seconds = 20.00
number of time cycles = 20000

number of Hydrophones = 3
distance from hydro 1 to hydro 0 = 1531.00
distance from hydro 2 to hydro 0 = 15310.00

Gaussian noise at hydros mean = 10.00
Gaussian noise at hydros SD = 5.00

multipath = 0
Number of targets = 2

average time between transient noises = 5.00
 the range to target 0 = 1531.00
  the angle for target 0 and the hydros = 60.00
  Broadband mean for target 0 = 1.00
```

```
 Broadband SD for target 0 = 2.00
Transient amplitude for target 0 = 20.00
 Number of frequencies at target 0 = 1
 Here are the frequency and amplitude pairs for target 0:
              40.00                  8.00

the range to target 1 = 1531.00
 the angle for target 1 and the hydros = 0.00
 Broadband mean for target 1 = 1.00
 Broadband SD for target 1 = 2.00
Transient amplitude for target 1 = 50.00
 Number of frequencies at target 1 = 2
 Here are the frequency and amplitude pairs for target 1:
             200.00                 10.00
              20.00                 20.00

Warm up takes 10000 time cycles
The simulation took 20.41 seconds.
```

Vita

Charles A. Jones was born on September 3, 1952, in Clarksburg, West Virginia. He received his Bachelor of Arts degree in Mathematics from Drake University, Des Moines, Iowa, in 1974. He received his Master of Arts degree in Mathematics from Dartmouth College, Hanover, New Hampshire, in 1976 and his Doctor of Philosophy in Mathematics from Dartmouth College in 1978. He has been employed by Grinnell College, Grinnell, Iowa, since 1980, first as an Assistant Professor of Mathematics, then as an Associate Professor of Mathematics.

MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October, 1994	Technical

4. TITLE AND SUBTITLE

Simulating a Multi-target Acoustic Array on the Intel Paragon

5. FUNDING NUMBERS

N00014-92-J-1303

N00014-93-1-0527

6. AUTHOR(S)

Charles A. Jones

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Center for Computational Statistics
George Mason University
Fairfax, VA 22030

8. PERFORMING ORGANIZATION REPORT NUMBER

TR no.108

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy
Office of the Chief of Naval Research
Mathematical Sciences Division
800 N. Quincy Street Code 1111SP
Arlington, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Navy position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This thesis is built around a parallel programming project which simulates the recordings of a linear array of hydrophones in the presence of several sources of acoustic signals (targets). Simulation of multi-target data with appropriate modeling of multipath, path refraction, and local and distant noise source is a useful and relatively sophisticated modeling chore. The signal suites include multiple frequency narrow band signals, broad-band flow noise, and randomly generated transient signals. The noise suites include coherent and incoherent Gaussian noise, impulsive noise, and continental shelf reverberation.This project is of interest from a parallel programming viewpoint because it uses the Paragon a true Multiple Intstruction Multiple Data (MIMD) machine with three types of nodes: 1)a Manager node, 2)target nodes, and 3)Hydrophone nodes. The target nodes are reponsible for generating the signal suites, transient noise, and the coherent noise. The hydrophone nodes are responsible for calculation of multipath, refraction and time delay as well as adding th local incoherent noise suite. While this simulation can be used as a stand-alone application, it also will form the foundation for a much larger, more sophisticated simulation, namely producing a virtual Command Information Center (CIC).(continuation of thi abstract is in this technical report p. v)

14. SUBJECT TERMS

parallel simulation, ocean noise model, target nodes hydrophone nodes

15. NUMBER OF PAGES

82

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October, 1994	Technical

4. TITLE AND SUBTITLE
Simulating a Multi-target Acoustic Array on the Intel Paragon

5. FUNDING NUMBERS
DAAL03-91-G-0039
DAAH04-94-G-0267

6. AUTHOR(S)
Charles A. Jones

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Center for Computational Statistics
George Mason University
Fairfax, VA 22030

8. PERFORMING ORGANIZATION REPORT NUMBER
TR no.108

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
This thesis is built around a parallel programming project which simulates the recordings of a linear array of hydrophones in the presence of several sources of acoustic signals (targets). Simulation of multi-target data with appropriate modeling of multipath, path refraction, and local and distant noise sources is a useful and relatively sophisticated modeling chore. The signal suites include multiple frequency narrow band signals, broad-band flow noise, and randomly generated transient signals. The noise suites include coherent and incoherent Gaussian noise, impulsive noise, and continental shelf reverberation.This project is of interest from a parallel programming viewpoint because it uses the Paragon a true Multiple Intstruction Multiple Data (MIMD) machine with three types of nodes: 1)a Manager node, 2)target nodes, and 3)Hydrophone nodes. The target nodes are reponsible for generating the signal suites, transient noise, and the coherent noise. The hydrophone nodes are responsible for calculation of multipath, refraction and time delay as well as adding the local incoherent noise suite. While this simulation can be used as a stand-alone application, it also will form the foundation for a much larger, more sophisticated simulation, namely producing a virtual Command Information Center (CIC). (continuation of this abstract is in this technical report p. v)

14. SUBJECT TERMS	15. NUMBER OF PAGES
parallel simulation, ocean noise model, target nodes hydrophone nodes	82
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL